USER'S MANUAL

SDK-386™*

* 386 is a trademark of Intel Corporation. Used by permission.

DRAFT

TABLE OF CONTENTS

PREFACE

SDK-386™*

MICROPROCESSOR DEVELOPMENT SYSTEM

* 386 is a trademark of Intel Corporation. Used by permission.

General

The 80386 chip is one of the most powerful microprocessors available in the market today. Contributing to its power are: a complete 32-bit architecture, a powerful instruction set, and on-chip support of paged virtual memory. In addition, the 80386 processor has a 4 gigabyte physical address space. The powerful 80386 microprocessor chip has been installed in the SDK-386™ single board microprocessor development system so that the user is able to utilize all of these assets.

This single board development system allows the user to edit programs, debug software, run programs and save the software on magnetic tape. In addition, the hardware is laid out in such a manner that permits expansion of the EPROM space, the RAM space, and the I/Ospace. This development system is the SDK-386™ .

General Specifications

- Complete Single Board Microcomputer System Including CPU, Memory and I/O
- A 80386 32 bit 16 (or higher) MHz microprocessor (Optional)
- 80387 Floating Point Coprocessor (Optional)
- 20 Mhz Clock
- Printed Circuit Card double sided, solder masked with silk screened component identification keyed to the schematic.
- A total of 25 integrated circuits (ICs).
- Sockets provided for critical ICs, e.g., processor, EPROM, RAM, I/O.

Memory

- 16 K bytes of EPROM with:

  7 K bytes of EPROM for Monitor and Utilities
  9 K Bytes of EPROM for User expansion

- 32 K bytes of static RAM

User Input/Output

- 5 by 8 Matrix Keypad (calculator type, 4 by 7 area used by the monitor, the 8 additional keys are available to the User along with a shifted positions on the Monitor scanned keys.)
- 40 character x 2 line Dot Matrix Liquid Crystal Display for display of addresses and data, messages, display of results, etc.
- Additional Keys:

  Hardware Reset Key
  Break Software Key
  User Interrupt Key

- Speaker for audio output

The SDK-386™ is an inexpensive microcomputer development system including power supply and instruction manual which is based on the 80386 microprocessor chip. The SDK-386™ is fabricated on a single printed circuit board and constructed from off-the-shelf components. The SDK-386™ is more competitive than other products because of the combination of inexpensive components from a variety of manufacturers emphasizing putting a low cost development system in as many hands as possible. The SDK-386™ is a product specifically designed for a specific market, i.e. educational (university, technical schools and even high schools), personal engineering systems, hobbyists, hackers, self study courses, etc.

Hardware Input/Output

- Two user 8 Bit parallel I/O Port
- Three 50 Pin DIP Connectors giving external access to all 80386 pins, I/O signals, and other functions.
- Input/Output provided through two 8255s.
- Hardware interface to inexpensive cassette tape recorder through two 1/8 inch phono jacks using MIC (microphone) and EAR (earphone) connections on the tape recorder.

Software

- The Software Monitor complete with listing is provided and includes convenient utilities to:

  Read/Scan the Keyboard
  Display data on the LCD display
  Tone generation
  Tape Write/Read routines

- Interrupt/Trap Vectoring
- Provide Break Points and Single Step Operation
- Provide Serial Communications with other computers
- Serial interface with user settable parameters

Power

- On board 5 volt, 1 amp (TTL) Power Capacity giving approximately 0.25 amp for User expansion.
- A 9 volt calculator type DC wall supply requiring 120 VAC/60 Hz.

Documentation

- User's Manual (approx. 60 pages) describing the system operation, key functions, examples, etc.
- Intel 80386 Programer's Reference Manual, detailing the 80386 processor, instructions, object (machine) codes, 80386 architecture, addressing modes, etc.
- Complete Schematic showing all components and interconnections

Optional Accessories

- Expansion Wire Wrap Prototype Board with Wire Wrap Header
- Expansion Ribbon Cables with Connectors
- Expansion Wire Wrap Kit including above, wire wrap tool, wire, sample circuits and components
- DRAM Expansion Card with up to 1 megabytes DRAM
- PASM-80386 Cross Assembler hosted on MS-DOS Computers

4516 Henry Street, Suite #407, Pittsburgh, PA 15213
1-800-338-0517 or 412-683-8732

FOR PRICES AND ORDERING INFORMATION, CALL:

URDA®, Inc.
University Research and Development Associates, Inc.
4516 Henry Street, Suite #407
Pittsburgh, PA, 15213
1-800-338-0517 or 1-412-683-8732

1.0 INTRODUCTION

1.1 General Description

The 80386 chip is one of the most powerful microprocessors available in the market today. Contributing to its power are: a complete 32-bit architecture, a powerful instruction set, and on-chip support of paged virtual memory. In addition, the 80386 processor has a 4 gigabyte physical address space. The powerful 80386 microprocessor chip has been installed in the SDK-386™ single board microprocessor development system so that the user is able to utilize all of these assets.

This single board development system allows the user to edit programs, debug software, run programs and save the software by uploading to a Personal Computer or outputting it to magnetic tape. In addition, the hardware is laid out in such a manner that permits expansion of the EPROM space, the RAM space, and the I/Ospace. Henceforth, this development system, will be referred to as the SDK-386™ .

1.2 Unpacking and Setting up the SDK-386™

Do not throw any of the packing material in the SDK-386™ away. The original carton and the other packing material can be used to store it, or ship it if necessary.

Remove the SDK-386™ from its shipping container, open it up, remove the documentation, the transformer and the packing material. You should now be able to place the SDK-386™ on your desk.

To power up the computer, plug the transformer into a source to 110 volt, 60 hertz power. Take the power jack on the cable attached to the transformer and plug it into the power receptacle.

The SDK-386™ should immediately power up and shortly after, the Liquid Crystal Display should show the message: SDK-386™ .

If you successfully obtain this message, the SDK-386™ is working properly. Please read the rest of the manual before proceeding.

1.3 Specifications

1. A 80386 32 bit 16 MHz microprocessor

2. 16 K bytes of EPROM with:

    a. 7 K bytes of EPROM for Monitor and Utilities
    b. 9 K Bytes of EPROM for User expansion

3. 32 K bytes of static RAM

4. 5 by 8 Matrix Keypad (calculator type, 4 by 7 area used by the monitor, the 8 additional keys are available to the User along with shifted positions on the Monitor scanned keys.)

5. 40 character x 2 line Dot Matrix Liquid Crystal Display for display of addresses and data, messages, display of results, etc.

6. Additional Keys:

    a. Hardware Reset Key
    b. Break Software Key

c. User Interrupt Key

7. 16.0 Mhz Clock

8. Two user 8 Bit parallel I/O Ports

9. Three 50 Pin DIP Connectors giving external access to 80386 pins, I/O signals, and other functions.

10. On board 5 volt, 1 amp (TTL) Power Capacity giving approximately 0.25 amp for User expansion.

11. Printed Circuit Card double sided, solder masked with silk screened component identification keyed to the schematic.

12. Complete Schematic showing all components and interconnections.

13. Speaker for audio output

14. Input/Output provided through two 8255s.

15. A total of 25 integrated circuits (ICs).

16. Sockets provided for all ICs, e.g., processor, EPROM, RAM, I/O.

17. A 9 volt calculator type DC wall supply requiring 120 VAC/60 Hz.

18. The Software Monitor complete with listing is provided and includes convenient utilities to:

    a. Read/Scan the Keyboard
    b. Display data on the LCD display
    c. Tone generation
    d. Tape Write/Read routines
    e. Interrupt/Trap Vectoring
    f. Provide Break Points and Single Step Operation
    g. Provide Serial Communications with other computers

19. Hardware interface to inexpensive cassette tape recorder through two 1/8 inch miniature phone jacks using MIC (microphone) and EAR (earphone) connections on the tape recorder.

20. User's Manual (approx. 60 pages) describing the system operation, key functions, examples, etc.

21. Intel 80386 Programer's Reference Manual, detailing the 80386 processor, instructions, object (machine) codes, 80386 architecture, addressing modes, etc.

22. Serial interfaces with user settable parameters

23. 80387 Floating Point Coprocessor Socket (Chip not included.)

24. Optional Accessories (continuously being developed):

    a. Expansion Wire Wrap Prototype Board with Wire Wrap Header
    b. Expansion Ribbon Cables with Connectors

c. Expansion Wire Wrap Kit including (1-2) above, wire wrap tool, wire, sample circuits and components

2.0 DESCRIPTION OF THE SDK-386™

The SDK-386™ is an inexpensive microcomputer development system including power supply and instruction manual which is based on the 80386 microprocessor chip. The SDK-386™ is fabricated on a single printed circuit board and constructed from off-the-shelf components. The SDK-386™ is more competitive than other products because of the combination of inexpensive components from a variety of manufacturers emphasizing putting a low cost development system in as many hands as possible. Traditionally, microprocessor manufacturers assemble such development systems using their own components for the corporate engineering market in which a unit price is not significant.

The SDK-386™ is a product specifically designed for a different market, i.e. educational (university, technical schools and even high schools), personal engineering systems, hobbyists, hackers, self study courses, etc., but which nevertheless will find an additional market in the single board computer market, O.E.M. sales, value added sales, etc.

This manual includes: (1) a description of the system architecture; (2) the circuit schematics; (3) the input/output methods; (4) the software monitor and operating system; and (5) an instruction manual with several programming examples.

3.0 HARDWARE

The hardware for the SDK-386™ was designed to minimize the number of components, reduce component costs by using only components which have multiple sources and to provide flexibility for expansion. The hardware design incorporates 25 integrated circuits, a Liquid Crystal Display and a keyboard composed of 3 individual keys and a 5 by 8 matrix keypad including unused keys for user expansion.

3.1 Power Supply

The SDK-386™ utilizes one regulated +5 volt supply . The average current required is 620 ma and the peak current is 750 ma.

The power supply is designed to handle 1 amp at 5 volts which allows approximately 250 ma for expansion devices. A 9 volt 1 amp unregulated wall mount supply provides the input to the L78T05 regulator. This regulator is rated 1 amp at 5 volts. A tantalum capacitor is placed between the regulator input and ground to prevent oscillations. The 100 uf capacitor is placed across the regulator output and ground to improve the transient response of the regulated supply. The negative terminal of this capacitor is considered the common ground from which the digital ground and the display ground branch. It is important to keep these two grounds separate since the high current fluctuations of the display can cause errors in the digital components if they share common lines. To provide further stability on the digital +5 and ground lines there is a .01 uf capacitor associated with most digital devices.

3.2 System Clock and IC Limitations

A 16.0 megahertz system clock is used in the SDK-386™ . This clock can be replaced by a slower clock if it is desired, but clock speeds slower then 6 MHz should not be used, as this is the minimum clock speed for which the 80386 microprocessor is typically rated. Faster clocks should not be used.

3.3 Memory - I/O Decoding

In order to minimize the hardware costs, the entire memory space of the 80386 is not decoded. It is however decoded in such a manner as to provide sufficient space for most hand assembled programs and to allow provisions for expansion and expansion systems. Only address lines A12 - A23 are used for decoding and therefore addressed locations greater than FFFFFFH will overlap the decoded area. The M/IO* control line is also used in the decoding, allowing for separate I/Oand memory spaces.

3.4 The SDK-386™ Memory Map

The SDK-386™ memory map is show in Figure 1. The SDK-386™ system memory includes EPROM and static RAM. An optional DRAM board is planned. Each section of the memory map is described in detail below.

3.4.1 Monitor / EPROMS Section Addresses 00FF8000 HEX through 00FFFFFF HEX are occupied by the SDK-386™ EPROMs. The SDK-386™ comes equipped with two 2764 EPROMS providing 16K bytes of Read Only Memory organized into a 16 bit bus. This EPROM memory "wraps around" after address 00FFBFFF HEX. Thus addresses 00FF8000 through 00FFBFFF are occupied by the same memory locations as addresses 00FFC000 HEX through 00FFFFFF HEX. The monitor program itself occupies addresses 00FF8000 HEX through 00FF9FFF HEX (and also 00FFC000 HEX through 00FFDFFF HEX), leaving addresses 00FFA000 through 0FFBFFF for use by the user. If desired, the two 2764 EPROMS may be removed from there sockets and replaced with two 27128 EPROMs which will eliminate the wrap around mentioned above and provide the user with a full 32K bytes of EPROM extending from address 00FF8000 HEX through 00FFFFFF.

Note: If the 2764 EPROMs are replaced with 27128s, as described above, the 27128 EPROMs must have an access time no greater than 200 ns. The monitor program must first be "burned" into the 27128s before they can be used in the SDK-386™ . This requires the use of a prom burner, such as PBURN µLAB™ available from URDA, Inc.

3.4.2 SRAM Section The SDK-386™ system is equipped with four 6264 (or equivalent) Static RAM (SRAM) ICs which supply it with 32K bytes of SRAM, organized into a 32 bit bus. The SRAM extends from address 00000000 HEX through address 00007FFF HEX. SRAM addresses 00000000 HEX through 000002FF HEX are reserved for system use by the SDK-386™ . All other SRAM locations are available for use by the user. The user stack begins at address 00007FE0 HEX and increases downward in memory. An optional DRAM expansion board is planned for RAM expansion.

3.4.3 DRAM Section Memory addresses 00008000 HEX through 00107FFF Hex are reserved for use by the optional DRAM extension board which is not included. The expansion board is connected to headers J1, J2 and J3 of the SDK-386™ board and allows the user to add up to 1 megabyte of additional memory to the system. When this section of the memory map is addressed by the 80386 processor, the DRAM chip select line is forced active (logical high). The hardware necessary to generate a READY to the processor is included on the DRAM expansion board. The DRAM signal may be used to interface user defined hardware to the SDK-386™ system if the DRAM expansion board is not used, however, the user hardware must also generate its own READY signal.

3.4.4 DRAM Bank 0 - Bank 3 As show in Figure 1 the DRAM section is divided into four banks, each containing 256K bytes of DRAM organized into a 32 bit bus. This allows the user to expand the system's memory in 256K increments, adding only as much additional RAM as needed. The address spaces of each bank of DRAM are as specified in Figure 1.

3.4.5 Unused Section Addresses 00108000 HEX through FFFF7FFF HEX are not used by the SDK-386™ system and are available for user expansion. The bus size is undefined in this area of the memory map and must be specified by generating the proper signal for the processor line BS16. An appropriate READY signal must also be generated.

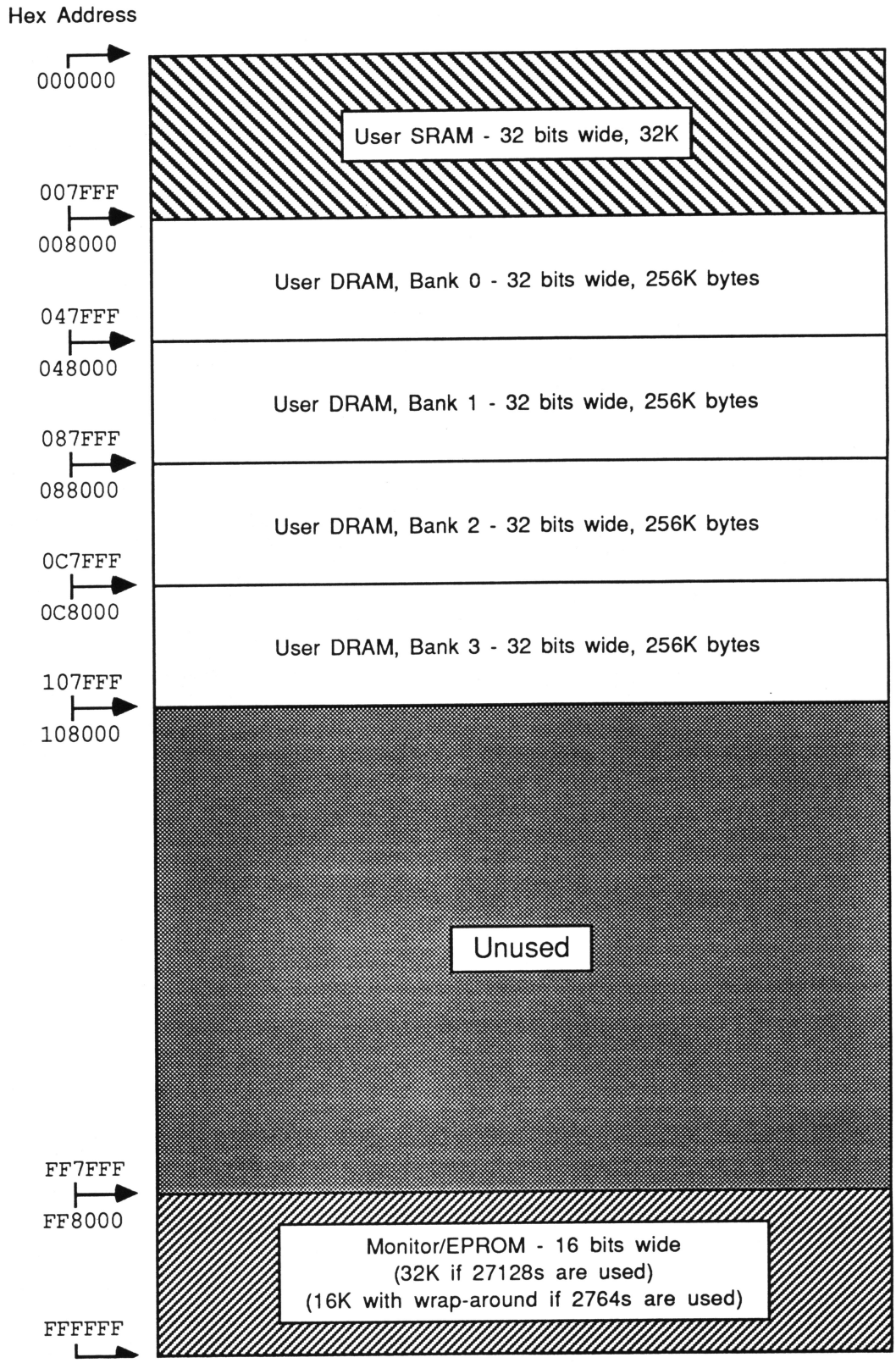


Figure 1. SDK-386™ Memory Map

3.5 The SDK-386™ I/OSpace

The SDK-386™ I/Ospace is distinct from the memory address space (i.e., it does not overlap with the memory address space). The total I/Ospace of the 80386 processor is 64K bytes. The I/Omap of the SDK-386™ is shown in Figure 2. Both parallel and serial I/Oare used in the SDK-386™ system. The 80386 IN and OUT instructions must be used to communicate with the SDK-386™ I/O.

3.5.1 Parallel I/OParallel I/Ois provided by two 8255 Programmable Peripheral Interface (PPI) ICs. These ICs use addresses E000 HEX through EFFF HEX of the 80386 I/Ospace.

3.5.2 Serial I/O Serial I/O is provided by the SDK-386™ 68681 Dual Universal Asynchronous Receiver / Transmitter (DUART). The SDK-386™ has two asynchronous serial channels which can be used to interface the SDK-386™ with other systems, such as a personal computer, as described in Section 5. The DUART uses addresses F000 HEX through FFFF HEX of the 80386 I/O space.

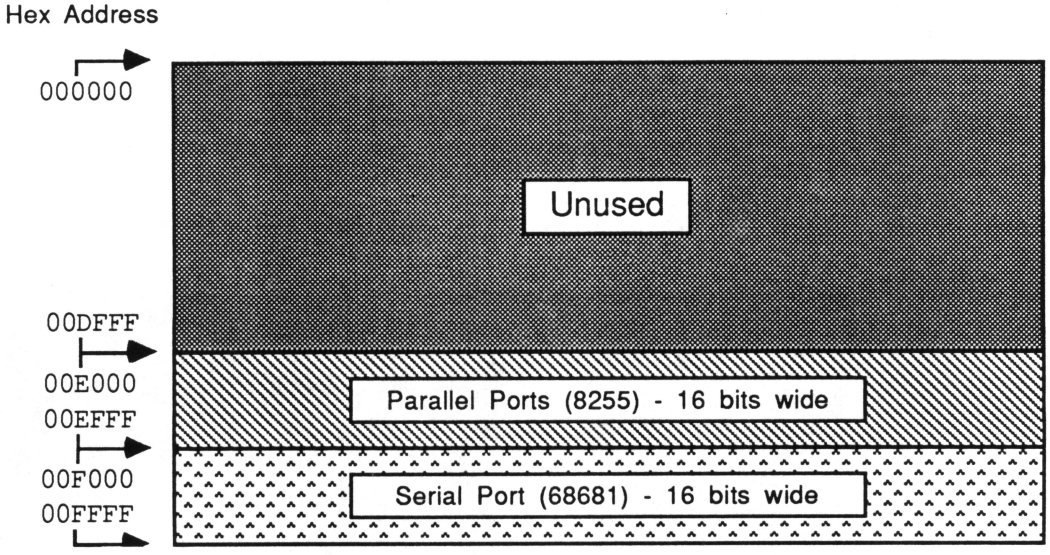


Figure 2. SDK-386™ I/O Map

3.6 Hardware Exceptions

The pushbuttons [BREAK], and [USER INT] are used to generate hardware exceptions (interrupts). [BREAK] generates a non-maskable interrupt (NMI) while [USER INT] generates a maskable interrupt (INT). The use of these pushbuttons is discused in more detail in Section 4.5

3.4 Keyboard

The keyboard is composed of a 5 by 8 matrix type keypad and three pushbutton keys. The [SHIFT] key is the first key detected. If multiple keys are pressed simultaneously then the last key detected in the scanning process is the one that is returned as the key pressed. The exception to this rule is the [SHIFT] key which is noted as being pressed along with the last key detected. See Section 4.0 for a complete description of key functions and Sections 5.3.1 and 5.3.2, respectively, for detailed descriptions of the Keyboard and Scan routines.

3.5 Display

The SDK-386™ system uses a 40 character, 2 line liquid crystal display to display addresses, data and messages. The display is accessed through the lower 8255 (IC 24). Refer to the schematic in Section 10.6 for complete details of the hardware connections between the 8255 and the display. Refer to Section 7.3 for a complete description of the monitor routines used for displaying data and messages. A list of hexadecimal display codes used to generate characters on the display is given in Table 6.

3.6 Reset Circuitry

The 80386 processor requires a delay after power-up of at least 15 CLK2 periods before the (RESET) line goes from the high state to the low state. This is accomplished through the use of the RC circuit shown in the schematic in Section 9.This causes the SDK-386™ to do system reset on power-up and go through it's initialization routines. Pressing the [RESET] pushbutton will also initiate a system reset, however, the actions taken by the SDK-386™ at power-up differ from the actions taken when the [RESET] pushbutton is pressed. Section 4.1.1 and 4.1.2 explain the differences between these two types of resets.

3.7 Parallel I/O - System 8255s

The SDK-386™ system utilizes two 8255 PPIs for input and output. The connections between the PPIs and peripheral devices are shown in the schematic diagram in Section 10.6. Table 2 gives the addresses of the various registers of the PIAs. Note that the two PIAs are organized into a 16 bit bus, so that both PIAs can be addressed an once using WORD instructions or either PIA can be addressed individually using a BYTE instruction. For complete information on programing these PIAs refer to an 8255 data sheet.

Register	Address - PIAL (IC 24)	Address - PIAH (IC19)
Port A	E000 HEX	E001 HEX
Port B	E002 HEX	E003 HEX
Port C	E004 HEX	E005 HEX
Control Register	E006 HEX	E007 HEX

Table 0

4.0 SOFTWARE

4.1 General Description of Monitor Function

The SDK-386™ system makes use of the 80386 processor's multitasking capabilities to run user written programs under the control of the monitor program. The monitor program contains the background routines which allow communication between the processor and the user. The monitor performs such functions as displaying messages, interpreting keyboard closures, acting upon keyboard closures, testing the RAM after power is applied and initializing the processor to a known state. The monitor program also handles many of the exception vectors which are possible with the 80386 microprocessor.

4.1.1 System Initialization: At power-up the SDK-386™ , under the control of the monitor program in EPROM begins by initializing the SDK-386™ system RAM. System RAM consists of memory

locations 00000000H through 000002FFH. Included in the system RAM are the Global Descriptor Table, the Local Descriptor Table, the Interrupt Descriptor Table, the System TSS and the User TSS. The locations of these data structures and the various fields within them are shown in Table 1.

Table 1-A GLOBAL DESCRIPTOR TABLE:

Field	RAM Address	Size (Bytes)
NULL DESCRIPTOR	00000000H	8
FLAT CODE DESCRIPTOR	00000008H	8
FLAT DATA DESCRIPTOR	00000010H	8
LDT DESCRIPTOR	00000018H	8
SYSTEM TSS DESCRIPTOR	00000020H	8
USER TSS DESCRIPTOR	00000028H	8
USER DEF. DESCRIPTOR	00000030H	8
USER DEF. DESCRIPTOR	00000038H	8
USER DEF. DESCRIPTOR	00000040H	8
USER DEF. DESCRIPTOR	00000048H	8

Table 1-B LOCAL DESCRIPTOR TABLE:

Field	RAM Address	Size (Bytes)
NULL DESCRIPTOR	00000050H	8
FLAT CODE DESCRIPTOR	00000058H	8
FLAT DATA DESCRIPTOR	00000060H	8
USER DEF.DESCRIPTOR	00000068H	8
USER DEF. DESCRIPTOR	00000070H	8
USER DEF. DESCRIPTOR	00000078H	8
USER DEF. DESCRIPTOR	00000080H	8
USER DEF. DESCRIPTOR	00000088H	8
USER DEF. DESCRIPTOR	00000090H	8

SYSTEM DATA **00000098H - 0000009FH**

Table 1-C INTERRUPT DESCRIPTOR TABLE:

Field	RAM Address	Size (Bytes)
ZERO DIVDE	000000A0H	8
DEBUG EXCEPT	000000A8H	8
NON MASK INT (BREAK KEY)	000000B0H	8
BREAK POINT	000000B8H	8
OVERFLOW	000000C0H	8
BOUNDS	000000C8H	8
INVALID OPCODE	000000D0H	8
COPROCESSOR NOT AVAL.	000000D8H	8
DOUBLE FAULT	000000E0H	8
COPROCESSOR SEG OVERRUN	000000E8H	8
INVALID TSS	000000F0H	8
SEG. NOT PRESENT	000000F8H	8
STACK EXCEPTION	00000100H	8
GEN PROTECTION ERROR	00000108H	8
PAGE FAULT	00000110H	8
RESERVED	00000118H	8

COPROCESSOR ERROR	00000120H	8
RESERVED	00000128H	8
RESERVED	00000130H	8
RESERVED	00000138H	8
RESERVED	00000140H	8
RESERVED	00000148H	8
RESERVED	00000150H	8
RESERVED	00000158H	8
RESERVED	00000160H	8
RESERVED	00000168H	8
RESERVED	00000170H	8
RESERVED	00000178H	8
RESERVED	00000180H	8
RESERVED	00000188H	8
RESERVED	00000190H	8
RESERVED	00000198H	8
USER INT. KEY	000001A0H	8
USER DEF. DESCRIPTOR	000001A8H	8

Table 1-D SYSTEM TSS:

Field	RAM Address	Size (Bytes)
SYSTEM BACKLINK	000001B0H	2
SYSTEM ESP0	000001B4H	4
SYSTEM SS0	000001B8H	2
SYSTEM ESP1	000001BCH	4
SYSTEM SS1	000001C0H	2
SYSTEM ESP2	000001C4H	4
SYSTEM SS2	000001C8H	2
SYSTEM CRS	000001CCH	4
SYSTEM EIP	000001D0H	4
SYSTEM EFLAGS	000001D4H	4
SYSTEM EAX	000001D8H	4
SYSTEM ECX	000001DCH	4
SYSTEM EDX	000001E0H	4
SYSTEM EBX	000001E4H	4
SYSTEM ESP	000001E8H	4
SYSTEM EBP	000001ECH	4
SYSTEM ESI	000001F0H	4
SYSTEM EDI	000001F4H	4
SYSTEM ES	000001F8H	2
SYSTEM CS	000001FCH	2
SYSTEM SS	00000200H	2
SYSTEM DS	00000204H	2
SYSTEM FS	00000208H	2
SYSTEM GS	0000020CH	2
SYSTEM LDT	00000210H	2
SYSTEM I/O MAP BASE	00000216H	2

SYSTEM DATA **00000218H - 0000021FH**

Table 1-E USER TSS:

Field	RAM Address	Size (Bytes)
USER BACKLINK	00000220H	2

USER ESP0	00000224H	4
USER SS0	00000228H	2
USER ESP1	0000022CH	4
USER SS1	00000230H	2
USER ESP2	00000234H	4
USER SS2	00000238H	2
USER CRS	0000023CH	4
USER EIP	00000240H	4
USER EFLAGS	00000244H	4
USER EAX	00000248H	4
USER ECX	0000024CH	4
USER EDX	00000250H	4
USER EBX	00000254H	4
USER ESP	00000258H	4
USER EBP	0000025CH	4
USER ESI	00000260H	4
USER EDI	00000264H	4
USER ES	00000268H	2
USER CS	0000026CH	2
USER SS	00000270H	2
USER DS	00000274H	2
USER FS	00000278H	2
USER GS	0000027CH	2
USER LDT	00000280H	2
USER I/O MAP BASE	00000286H	2

GDT POINTER	00000288H	6
IDT POINTER	00000290H	6

SYSTEM DATA **00000298 - 000002FF**

Table 1. System RAM

4.1.2 Global Descriptor Table The Global Descriptor Table extends from location 00000000H through location 0000004FH and has space for ten 8 byte descriptors. Only the first five descriptors are used by the SDK-386™ system monitor however, and the rest may be defined by the user. The Flat Code Descriptor and Flat Data Descriptor are used by the monitor program for it's code and data segments respectively. These descriptors define segments for the "flat" system model and extend the entire memory space of the 80386 processor (00000000H through FFFFFFFFH). A selector for the Flat Code Descriptor is also loaded into the CS field of the User TSS and a selector for the Flat Data Descriptor is loaded into the ES, SS, DS, FS, and GS fields of the User TSS. The code and data segments for the user task may be changed by loading an appropriate descriptors in the user definable fields of the Global (or Local) Descriptor Tables and then loading an appropriate selector into the User TSS segment register fields.

4.1.3 Local Descriptor Table The Local Descriptor Table extends from location 00000050H through location 00000097H and has space for nine 8 byte descriptors. None of the local descriptors is used by the SDK-386™ system monitor and all of them may be redefined by the user except the first one, the Null Descriptor, which is required by the 80386 processor. The second and third descriptors are the Flat Code Descriptor and Flat Data Descriptor which may be redefined by the user.

4.1.4 Interrupt Descriptor Table The Interrupt Descriptor Table extends from location 000000A0H through location 000001BFH and has space for thirty four 8 byte descriptors. Most of these descriptors are reserved for exceptions and error conditions encountered by the 80386 processor. Descriptor 32 is for a level 32 non-maskable interrupt, which is generated by the

[USER INT] pushbutton key and descriptor 33 is user definable. For additional information on using the Interrupt Descriptor Table see Section 7.1.

4.1.5 System Task State Segment: The System Task State Segment is used by the system task under which the monitor program runs. As part of the initialization procedure, the monitor program loads a selector for the User Task Descriptor (the sixth descriptor in the Global Descriptor Table), which points to the User TSS, into the processor's LTR register. The monitor then calls the system task. This causes the 80386 processor to record the selector for the User Task Descriptor in the backlink field of the System TSS. When the user then presses the [RUN] or [STEP] key, an IRET instruction is preformed, causing the 80386 processor to return to the user task and begin running the user's program. If the [BREAK] pushbutton key is pressed, a non-maskable interrupt is generated, which calls the system task again. In this way the user may run user programs and return to the monitor program at any time by pressing the [BREAK] key. None of the fields in the System TSS should by modified by the user or the monitor program may not work correctly. For additional information on running user programs see Section 4.3.

4.1.6 User Task State Segment: The User Task State Segment is used by the user task which runs user programs. The method by which user programs are run was described in Section 4.1.5. Any of the fields in the User TSS may be modified by the user. In addition, the monitor provides the user with the [R32] and [R16] keys (see Section 4.3.1) which allow the user to view and change any of the register fields in the User TSS directly.

4.1.7 System State After Warm Restart A warm restart is initiated by pressing the [RESET] pushbutton key, which generates a processor reset. Pressing the [RESET] key causes the monitor program to reinitialize the first six descriptors of the Global Descriptor Table and the System TSS. All other system RAM retains the value it had before the [RESET] key was pressed.

The warm restart capability retains the user entries in the RAM locations. Thus, if a user program gets "lost", the RESET button gives control without causing the user to reenter all of the program/data.

The 80386 is a sophisticated microprocessor with various table requirements. A "lost" program can destroy user program/data entries that may go unnoticed with a warm restart. A cold restart can easily be effected by changing the byte in Monitor/EPROM memory location 00FF80F1H from its current "warm start" value of 55H to any other value.

If the user does not have the capability to burn EPROMs, simply mail the "warm start" EPROMs to URDA, Inc., and a "cold start" set of EPROMs will be returned to the user.

4.2 Entry of Programs via the Keyboard

The main purpose of certain keys is for entry, editing and execution of user programs. A list of Key Codes and Key Positions is given in Table 5. There are three major keys whose functions must be clearly understood before proceeding.

4.2.1 Shift, Break and Reset Keys Most of the keys on the keyboard have dual functions. The [SHIFT] key is used to access the second function of the keys. Pressing the [SHIFT] key causes the message **Shift** to appear on the display, indicating that the shift function is active. Pressing it a second time clears the display and nullifies the shift function. The shift function is only active for the key press immediately following the closure of the [SHIFT] key and therefore the [SHIFT] key must be pressed before each key which requires a second function.

The Break [BREAK] key is provided to allow the user to interrupt an executing program to check the current location of the instruction pointer. Pressing the [BREAK] key causes the display to show the current value of the user instruction pointer and the data contained in that memory location.

This key thus provides a hardware program break by virtue of time (key closure) as opposed to a software break by virtue of an address. This function is useful in program debugging to find a trapping loop location(s).

The user can continue execution of the interrupted user program by pressing the run [RUN] key (Section 4.3.5).

The break [BREAK] key generates a non-maskable interrupt (NMI). Pressing this key while the monitor program is running will have no effect on the SDK-386™ system.

The reset [RESET] pushbutton is a SPST switch connected to a pull up resistor on one terminal and ground on the other terminal. The pull up terminal is hard wired through a charging circuit to the (RESET) pin on the 80386 chip.

Pressing this key asserts a reset throughout the SDK-386™ system with the processor going through the reset function described in Section 4.1.2.

4.2.2 Base Sixteen Numeric Keys There are 16 numeric keys located on the right half of the keyboard. These keys are the hexadecimal program and data keys. Their use does not require the [SHIFT] key and their action is dependent on the mode of the monitor routine at the time of closure. Generally, they are displayed at the time of closure.

4.2.3 Address Key The [ADDR] key is used to set a memory address. On power-up the value of the address variable is set to 00000300H.

After pressing the address key [ADDR], the address contained in the address variable will be displayed in the left eight characters of the display, with a left pointing arrow as the seventh character. The hexadecimal byte that is at that address is indicated by the right two characters on the display. The arrow, pointing towards the address indicates the portion of the display field which can now be modified with the hexadecimal keys. Pressing any hexadecimal key after pressing the address key causes the digit pressed to be placed in the least significant digit of the address. Each additional key pressed also enters the display as the least significant digit by being shifted right to left on the display.

Note: The address will only be six digits long. The high order two digits are always set to zero by the monitor since the address space greater than 00FFFFFF HEX is not decoded. As each hexadecimal key is pressed, the byte value at the indicated location will be displayed.

As an example, suppose the [ADDR] key is pressed right after power-up. The display will be: **00000300<-xxxxxx00** , indicating address 00000300 HEX contains the data 00 HEX. Next the [4] key is pressed. The display will be: **00003004<-xxxxxx00**, indicating address 00003004 contains the data 00 HEX. If next the [0] key is pressed, the display will be **00030040<-xxxxxxxx** indicating that location 000400 HEX is beyond user RAM, and cannot be accessed through the monitor program.This process can be continued until the left four digits display the desired address.

4.2.4 Data Byte, Data Word and Data Long Word Keys The data byte [DAB], data word [DAW] and data long word [DALW] keys all place the monitor in the data entry mode. After any of the data keys is pressed the left eight digits of the display will contain the address (see Section 4.2.3 for selecting the address), followed by a right pointing arrow, indicating that the data field can be altered. If the [DAB] key was pressed, then the arrow will be followed by six lower case Xs and then two digits (the byte value at that address). If the [DAW] key was pressed, then the arrow will be followed by four lower case Xs and then four digits (the word value at that address). If the [DALW] key was pressed, then the arrow will be followed by eight digits (the long word value at that address).

The hexadecimal keys function in the data entry mode similar to their function in the address mode. The entered digit is shifted left into the least significant digit of the display. Each

subsequent hexadecimal digit is simply shifted left into the least significant digit with the most significant digit being discarded.

For example, suppose the address 000310 HEX has been entered as described in Section 4.2.3. The display will be: **00000310<-xxxxxx00**. Pressing the [DAW] key next, will change the display to: **00000310->xxxx0000** then [5]: **00000310->xxxx0005** then [6]: **00000310->xxxx0056 then** [7]: **00000310->xxxx0567** then [8]: **00000310->xxxx5678** and then [9]: **008010->xxxx6789**. If the [DAB] key is then pressed the display will change to: **00000310->xxxxxx89**. At this point address 0310H will contain 89 Hex and address 0311 Hex will contain 67 Hex.

4.2.5 Plus Key The plus [+] key is a multifunction key. In general, it can be thought of as being similar to an enter key. When in the data mode, as described in Section 4.2.4, the plus key causes the data on the display to be stored in the memory location that is on the display. The monitor then advances the displayed address to the next address. When in the byte mode the address is incremented by one, when in the word mode the address is incremented by two and when in the long word mode the address is incremented by four.

If the monitor was in the address mode (Section 4.2.3), then the address is incremented by one and the monitor is placed in the byte data entry mode (Section 4.2.4). Thus, pressing the plus key while in the address mode, is equivalent to pressing the [DAB] key followed by the [+] key.

The [+] key is conveniently used for entering short segments of program or data. Enter the data using the hexadecimal keys (word or byte depending on the mode selected), press the [+] key, and repeat the process until all of the data are entered.

The other aspects of the [+] key will be described in detail in the sections that follow.

4.2.6 Minus Key The minus key [-] functions identically to that of the plus [+] key described in Section 4.2.5 except that, it causes the address to be decremented rather than incremented. The new address equals the old address minus one if in the byte mode, minus two if in the word mode and minus four if in the longword mode. The [+] and [-] key can be used for viewing sections of memory. Select a starting address as described in Section 4.2.3 and then select either byte, word or long word mode as described in Section 4.2.4. The contents of higher addresses can be viewed by using the [+] key, or lower addresses by using the [-] key. Additional features of the [-] key will be described in the sections to which they pertain.

4.2.7 Autoload Key The Autoload key [AUTO] places the monitor in a mode which saves the user one key stroke for each byte, word or long word entered. Prior to pressing the [AUTO] key, the user should select either the byte, word or long word mode as described in Section 4.2.4. If the word mode is selected, then after pressing the [AUTO] key, the monitor will store the data and increment the address by two after every four hexadecimal digit entries. The monitor will then display the contents of the next location. Pressing any key other than the hexadecimal keys will remove the monitor from the autoload mode. If the byte mode was selected, then the monitor will increment the address by one after every two hexadecimal digit entries. If the long word mode was selected, then the monitor will increment the address by four after every eight hexadecimal digit entries. This mode is convenient for entering large amounts of data or program into the system RAM.

4.2.8 Relative Key The Relative [REL] key calculates the relative offset from the current address to the destination address. This routine simplifies the calculation of the relative offsets for branch instructions. The data mode also plays a key role in this routine. If the byte mode was selected prior to pressing the [REL] key, then the Relative routine will calculate the byte offset. If the offset is too large for a single byte, then the monitor will display **Offset Too Large**. If the monitor was in the data word mode, then the offset will be calculated as a word. If the monitor was in the data longword mode, then the offset will be calculated as a longword. Once again, if the offset is too large to be a word, then the monitor will display **Offset Too Large**.

To utilize the relative function, enter the hexadecimal code for the branch instruction, advance to the next address (this will be done for the user if in the autoload mode), and press the [REL] key. The monitor will display **Dest----->**, indicating that a destination address should be entered. The address displayed to the right of this prompt is the last destination address entered. The user should then key in the hexadecimal address to which program execution is to branch, and then press the [+] key.

Note: In this case the [+] is used as an enter key. If the address is within range of the data mode selected, the monitor will calculate the offset, and load it into the proper address (a byte if in byte mode and a word if in the word mode). The monitor will then display the address with the calculated offset as the data.

Note: If the user was in the autoload mode, then after the relative address calculation, the user should advance to the next address and press the [AUTO] key to return to the autoload mode.

4.2.9 Move Key The Move [MOV] key can be used to move a block of code from one memory location to another. After pressing the [MOV] key, the monitor will display: **Start----->,** which is the prompt for the starting location. This address should be the low order address of the block of code to be moved. The user would then key in the hexadecimal starting address, and then press the enter [+] key. The monitor then determines if the address is within user RAM. If it is not, the monitor will display the error message: **Outside User RAM**. If the address is within user RAM, then the monitor prompts for the ending address of the block to be moved with the message: **End ----->**. The user should then key in the address of the last byte in the block to be moved plus one, and then press the enter key [+]. The monitor then checks if this address is within user RAM as it did with the starting address. If the address is in user RAM, then the monitor prompts for the address to which the block is to be moved with the display: **Dest----->**. This address should be the lowest order address of the destination. The user should then key in the hexadecimal address of the destination and then enter the address with the [+] key. The monitor also checks this address for its presence in user RAM. If the destination address is within user RAM, the monitor then determines if the last address of the destination is within user RAM. If it is, then the data are copied to the destination. If the starting address is less than the destination address then, the data are copied from the lower order address through the high order address. If the starting address is higher than the destination address, then the data are copied from the high order address through the low order address.

The minus key [-] also has an additional function in the Move routine. The [-] key enables the user to go back to the previous entry. For example, if the display is: **Dest----->**, pressing the [-] key will change the display to: **End ----->**, thus permitting the user to modify the ending address. If the [-] key is pressed when the monitor is requesting the starting address: **Strt----->**, the monitor will display the current address and data as it was before the MOV key was pressed. The MOV function has now been terminated and the user may select a new function key. In every case where the monitor is displaying a prompt and an address, the address displayed is the last address the user entered for the particular prompt.

4.2.10 Insert Key Data or program code may be inserted through the use of the insert [INS] key. Set the display to the address at which the insertion is to be made as described in Section 4.2.3. Set the mode to the data size of the insertion (byte,word or long word). Press the [INS] key. The monitor will when shift the contents of the displayed location and all of the data in every higher location up either one, two or four locations, depending on the data mode selected. The last location that the data are shifted into is the top of program RAM that was described in Section 4.1.1. The data that were previously in this location will be lost. Thus, variables which the user might wish to remain unaffected by insertion or deletion should be in user variable RAM.

Note: The monitor does not clear the data from the address of the insertion, but merely performs a Move with the destination address being either the current address plus one, two or four, depending on the data mode selected. The ending address is set to the top of program RAM, minus one or two (00007E00).

4.2.11 Delete Key Data or program code can be deleted through the use of the delete [DEL] key. Set the display to the address at which the data are to be deleted as described in Section 4.2.3. Set the mode to the data size of the deletion (byte or word). Press the [DEL] key. The monitor deletes the entry by performing a MOVE with the starting address set equal to the displayed address plus one in byte mode, plus two in the word mode or plus four in the long word mode. The destination address is the displayed address, and the end address is the top of program RAM. The delete routine thus follows the same rules that apply to the [INS] and the [MOV] keys.

4.3 Running and Debugging Software

There are several keys which select routines that can be helpful in debugging software. These keys permit the user to execute code, stop execution, and review or modify the processors registers.

4.3.1 Register Keys The [R32] and [R16] keys permit the user to review and change the contents of the register fields of the user TTS. To select a 32 bit register, begin by pressing the [R32] key. The display will show the message **CR3-----< XXXXXXXX** indicating that the CR3 register contains the HEX value XXXXXXXX. Note that the arrow is pointing toward the register name, indicating that the register field may be changed. Other 32 bit registers may now be selected by pressing the key which corresponds to the desired register. The 32 bit register keys are shared by the HEX keys 0 - A and the 32 bit register to which each key corresponds is shown in the center of the key. To change the contents of a register press the [INS]. the arrow will now point to the right, indicating that the data field may be changed. A new value may now be entered into the register using the HEX keys in the usual manner. When the correct HEX value appears in the data field of the display, press the [+] key. The arrow will point to the left again, and a new 32 bit register may be selected. To leave the [R32] function press the [+] key again. The monitor will display the current address and data in the usual manner.

The [R16] operates in a manner similar to the [R32] key except that it is used to view and change the data of 16 bit registers. The [R16] key is a second function key, and the [SHIFT] key must be pressed before pressing the [R16] key. To select a 16 bit register, begin by pressing the [SHIFT] key followed by the [R16] key. The display will show the message **ES------< xxxxXXXX** indicating that the ES register contains the HEX value XXXX. Note that the arrow is pointing toward the register name, indicating that the register field may be changed. Other 16 bit registers may now be selected by pressing the key which corresponds to the desired register. The 16 bit register keys are shared by the HEX keys 0 - 6 and the 16 bit register to which each key corresponds is shown in the top of the key. To change the contents of a register press the [INS]. The arrow will now point to the right, indicating that the data field may be changed. A new value may now be entered into the register using the HEX keys in the usual manner. When the correct HEX value appears in the data field of the display, press the [+] key. The arrow will point to the left again, and a new 16 bit register may be selected. To leave the [R16] function press the [+] key again. The monitor will display the current address and data in the usual manner.

4.3.2 Break-point Key The Break-point [BRPT] key can be used to set a break point which will stop program execution at a specific RAM address. The enter the break point edit mode, begin by pressing the [BRPT] key. The display will show the message **BrkPt0-C>XXXXXXXX** indicating that break point 0 may be set. XXXXXXXX indicates the address last entered for break point 0. This address may be changed using the HEX keys in the usual manner. The C to the left of the arrow indicates that the break point is currently cleared. To stop program execution at the address entered for the break point, the break point must be set. this is done by pressing the [SET] key. The display will now show the message **BrkPt0-S>XXXXXXXX** indicating that the break point is set and that program execution will terminate when the instruction pointer points to this address. Note that the [SET] and [CLEAR] keys are only used while the user is in the break point edit mode. They have no function and are not recognized by the monitor at any other time. They are also not second function keys, and the [SHIFT] key should not be pressed before pressing the [SET] or [CLEAR] key. The break point may be cleared again by pressing the [CLEAR] key.

The SDK-386™ allows the user to set up to four break points labeled 0,1,2,3. The user may view and change the address and status of the next break point by pressing the [+] key. The previous break point may be reviewed or changed by pressing the [-] key. In this manner, all four break points may be reviewed or changed by using the [+] and [-] keys. Break point edit mode is exited by pressing the [+] key when the status of break point 3 is displayed or pressing the [-] key when the status of break point 0 is displayed.

4.3.3 Step Key The step key [STEP] permits the user to execute a single instruction and then return to the monitor to view or modify registers and/or data. The [STEP] key can be used for stepping through software that is either in RAM or ROM since it does not entail any modification of the user code. Stepping is performed by the monitor by setting the TF bit in the flags register.

4.3.4 Run Key The run [RUN] key causes monitor program to perform a task switch to the user task. Instruction execution then begins at the address contained in the EIP field of the user TSS. The display will be blank unless the code being executed utilizes the display. Execution will continue until an exception occurs or the processor is reset.

4.4 Program Storage and Retrieval on Magnetic Tape

The SDK-386™ is equipped with the hardware and software, necessary for magnetic tape storage and retrieval of programs and data. Each bit is transmitted from the SDK-386™ to the tape as three cycles of a particular frequency square wave. When the 80386 is reading the data from tape, the period of the square wave is used to determine whether the bit is a one or a zero.

The format for a block of data is as follows. A leading string of 500 long periods is first sent. The leading string is immediately followed by four bytes for the filename, four bytes for the starting address, four bytes for the ending address and four bytes for the check sum.

The check sum is the long word result of the sum of all of the data that are transferred. The check sum is used for transmission error detection. After the check sum, a string of 500 short periods followed by sixteen long period is sent to signal the beginning of the actual data. This string is immediately followed by the actual data.

There are two miniature phone jacks located on the SDK-386™ that are to be used for the tape interface. The jack labeled EAR should be connected to the earphone jack of the tape recorder, and the jack labeled MIKE should be connected to the microphone jack of the tape recorder.

Note: The quality of the tape and the tape recorder can affect the accuracy of data storage and retrieval. The design of the interface is such that only moderate quality is required and therefore most general purpose cassette recorders can be used.

4.4.1 Tape Write Key The tape write [TPWR] key is used to transfer data from memory to magnetic tape. Prior to performing a tape write, the jack labeled MIKE on the SDK-386™ should be connected to the microphone jack of the tape recorder. Pressing [SHIFT] [TAPW] will cause the monitor to display the message: **File----->**, which is the prompt for the filename of the data to be transferred, followed by eight hexadecimal numbers indicating the previously entered filename. At this time, any eight digit hexadecimal number can be entered for the filename. The digits entered are rolled in and out of the display from right to left as with data and address entries.

When the desired filename is on the display, pressing the [+] key will enter the displayed value as the filename. The monitor will then prompt with the message: **Strt----->**, for the starting address of the block of data to be transferred to tape. Once again the hexadecimal digits following the prompt represent the last starting address entered. This address can be altered using the hexadecimal keys in the same manner as the filename.

When the proper starting address is displayed, pressing the [+] key will enter the displayed address as the starting address of the block of data to be transferred to tape. The monitor will then prompt, with the display: **End ----->**, for the ending address of the block of data. The displayed address is the last address from which the data are to be transferred to tape. Alteration of this address is the same as for the starting address.

Prior to pressing the [+] key to enter the ending address, the tape recorder should be turned on in the recording mode.

Immediately after pressing the [+] the monitor begins writing the selected block of data to the tape recorder. Upon the completion of data transfer, the display will be that of the last transferred address.

If it is desired to abort the transfer of data to tape while data are being sent to the tape, the reset key must be used since the monitor is dedicated to the data transfer and is not scanning the keyboard. When entering the starting address or ending address, the [-] key can be used to go back to the previous entry. If the [-] key is pressed while the monitor is prompting for the filename, the monitor will display the current address and data in the usual manner, as that were before the [TPWR] key was pressed.

Since the programs or data written to tape are placed in the same addresses when read back from tape, only addresses contained in user RAM are considered valid. If the starting address or ending address is outside of user RAM, the corresponding entry will cause the monitor to display the message **Outside User RAM**.

4.4.2 Tape Read Key The tape read [TPRD] is used to retrieve programs or data from magnetic tape. After pressing the [TAPR] key, the monitor will display: **File----->**. This message is prompting for the filename. The hexadecimal value following the prompt is that of the previously entered filename. This filename should be modified to that of the file desired.

Prior to pressing the [+] key for enter, the earphone jack of tape player must be connected to the EAR jack of the SDK-386™ , and the tape player must be turned on in the play mode. It is recommended that the volume control on the tape player be turned all the way to full volume.

Immediately after pressing the [+] key the monitor will clear the display. When a filename is found, the monitor will show it on the display. If this is the filename desired, the display will load the file into RAM. If the file is other than the one desired, the monitor will continue looking for the desired file.

After loading the selected file into RAM the monitor checks the check sum to determine if there were any transmission errors. If there were none, the monitor will display the address and data of the last byte received. If a transmission error occurred, the monitor will display the error message: **File Error**.

It is recommended that the reset key be used to abort the tape read routine. Only the reset and interrupt keys are active during a tape read since the monitor is dedicated to reading the tape input and does not scan the keyboard.

Note: The tape storage hardware and software are provided. However, URDA can not support the operation other than to insure the hardware is as indicated on the schematic and the software executes as indicated in the Monitor Listing. The storage of data is best accomplished by uploading to a Personal Computer through the serial port and then saving on a disk. The upload/download through the serial port is fully supported by URDA, Inc.

4.5 User Definable Keys

One of the pushbuttons on the SDK-386™ and eight of the keys on the keypad have no function in the monitor. These keys are provided for use by the user. Their use is described below.

4.5.1 User Interrupt Key The user interrupt key [USER INT] generates a level 32 maskable interrupt. To use this key, the user must provide a suitable interrupt gate in the SDK-386™ Interrupt Descriptor Table for a level 32 exception (memory locations 000001A0H through 000001A7H). The user must also enable maskable interrupts by setting the IF flag in the 80386 processor's EFLAGS register. This can be done using the STI instruction. When interrupts are enabled, the SDK-386™ will execute the interrupt handler routine pointed to by the level 32 exception descriptor when the [USER INT] key is pressed.

cause the interrupt vector, 32H, to be placed on the data bus. Thus, the [USER INT] key should only be pressed when it is desired to produce a maskable interrupt or to place the byte 32H on the data bus.

Note: A suitable and correct interrupt gate must be entered into the Interrupt Descriptor Table for a level 32 exception (memory locations 000001A0H through 000001A7H) before interrupts are enabled or the SDK-386™ will not function correctly when the [USER INT] key is pressed.

4.5.2 Other User Definable Keys Keys [U0] through [U7], as well as their shifted functions [SU0] through [SU7] of the key pad are all user definable keys which are already hardware decoded but are not implemented in the operating system. These keys are user programmable and can be set by the user to perform his/her own functions (see Section 7.3.2).

4.6 Alpha-Numeric List of Keys

KEY NAME	SECTION	DESCRIPTION	NOTES
+	4.2.5	PLUS KEY	(1,3)
-	4.2.6	MINUS KEY	(1,3)
0	4.2.2	BASE SIXTEEN NUMERIC	(1)
1	4.2.2	BASE SIXTEEN NUMERIC	(1)
2	4.2.2	BASE SIXTEEN NUMERIC	(1)
3	4.2.2	BASE SIXTEEN NUMERIC	(1)
4	4.2.2	BASE SIXTEEN NUMERIC	(1)
5	4.2.2	BASE SIXTEEN NUMERIC	(1)
6	4.2.2	BASE SIXTEEN NUMERIC	(1)
7	4.2.2	BASE SIXTEEN NUMERIC	(1)
8	4.2.2	BASE SIXTEEN NUMERIC	(1)
9	4.2.2	BASE SIXTEEN NUMERIC	(1)
A	4.2.2	BASE SIXTEEN NUMERIC	(1)
ADDR	4.2.3	ADDRESS KEY	(1)
AUTO	4.2.7	AUTOLOAD KEY	(1,3)
B	4.2.2	BASE SIXTEEN NUMERIC	(1,3)
BREAK	4.2.1	MONITOR KEY	(1,3,4)
BRBP	4.3.2	BREAK POINT KEY	(1)
BYTE	4.2.4	BYTE KEY	(1,3)
C	4.2.2	BASE SIXTEEN NUMERIC	(1,3)
CLEAR	4.3.2	CLEAR BREAK POINT KEY	(2)
D	4.2.2	BASE SIXTEEN NUMERIC	(1,3)
DWD	4.2.4	DOUBLE WORD KEY	(1,3)
DEL	4.2.11	DELETE KEY	(2)
DNLD	5.5	DOWNLOAD KEY	(1)
E	4.2.2	BASE SIXTEEN NUMERIC	(1)
F	4.2.2	BASE SIXTEEN NUMERIC	(1)
INS	4.2.10	INSERT KEY	(1)
MOV	4.2.9	MOVE KEY	(1,3)
REL	4.2.8	RELATIVE KEY	(2)
RESET	4.2.1	RESET KEY	(1,3)
RUN	4.3.5	RUN KEY	(1)
SET	4.3.2	SET BREAK POINT KEY	(2)
SHIFT	4.2.1	SHIFT (SECOND FUNCTION) KEY	(1,3)
STEP	4.3.4	STEP KEY	(1,3)
SU0	4.5	USER KEY	(2,5)
SU1	4.5	USER KEY	(2,5)
SU2	4.5	USER KEY	(2,5)
SU3	4.5	USER KEY	(2,5)
SU4	4.5	USER KEY	(2,5)
SU5	4.5	USER KEY	(2,5)
SU6	4.5	USER KEY	(2,5)
SU7	4.5	USER KEY	(2,5)
TAPR	4.4.2	TAPE READ KEY	(2)
TAPW	4.4.1	TAPE WRITE KEY	(2)
U0	4.5	USER KEY	(1,5)
U1	4.5	USER KEY	(1,5)
U2	4.5	USER KEY	(1,5)
U3	4.5	USER KEY	(1,5)
U4	4.5	USER KEY	(1,5)

U5	4.5	USER KEY	(1,5)
U6	4.5	USER KEY	(1,5)
U7	4.5	USER KEY	(1,5)
USER INT	4.5	USER INTERRUPT KEY	(1,3,4,5)
UPLD	5.4	UPLOAD KEY	(1)
WORD	4.2.4	WORD KEY	(1,3)

NOTES:

(1) First Function
(2) Second Function
(3) Single Function Key
(4) Separate Pushbutton to Left of 5x8 Keyboard
(5) User Definable Key

5.0 SERIAL COMMUNICATIONS

5.1 Serial Port Specifications

The SDK-386™ is equipped with the 68681 Dual Asynchronous Receiver/Transmitter (DUART) and software driver utilities to enable it to communicate with other computers and systems through a serial port. The default parameters of the SDK-386™ serial port are:

(a) 8 Data Bits;
(b) 2 Stop Bits;
(c) No Parity;
(d) 9600 Baud.

The serial communications port parameters given above can be altered by modifying the software driver program in EPROM or with a user written serial driver program residing in RAM. The utilities provided with the SDK-386™ monitor allow the user to transfer files (containing data or programs) between the SDK-386™ and computer such as the IBM PC or AT. Transferring a file from the SDK-386™ to another computer is referred to as an Upload operation while transferring a file from another computer to the SDK-386™ is referred to as a Download operation.

5.1.1 Hardware Connections An appropriate cable should be connected between the DB25 connector on the SDK-386™ and the serial port of the computer with which the SDK-386™ will communicate. The pin-outs of the SDK-386™ DB25 serial port connector are given in Section 10.4. In addition, complete cabling diagrams are given for the IBM PC® and Apple McIntosh® computers in Section 9.0.

5.2 Transferring Files Between the SDK-386™ and Another Computer

The SDK-386™ Upload/Download monitor software utilities (resident in the SDK-386™ EPROMs) allow the SDK-386™ to send and receive files through the serial port of another computer as a sequence of 8 - bit characters. When a file is transferred from the SDK-386™ to another computer, the transfer is referred to "up-loading" and hence, the SDK-386™ Upload utility is used. When the direction is reversed, and a file is transferred to the SDK-386™ from another computer, the transfer is referred to as "down-loading" and the SDK-386™ Download utility is used.

While the SDK-386™ Upload/Download utilities handle the transfer of files at the SDK-386™ end of this communications link, the other computer must also have software that allows it to send and receive files as well to complete the link. The SDK-386™ comes supplied with a

software package for linking the SDK-386™ to the IBM-PC® or AT® microcomputers. The software for linking to the IBM-PC® or AT® microcomputers is called PCOM-PC and is supplied on a 5 1/4" floppy disk. The use of PCOM-PC is covered in Section 6.0. It possible to link the SDK-386™ to other computers, however software for such machines is not supplied with the SDK-386™ and must be supplied by the user. The remainder of this section will be devoted to the operation and use of the SDK-386™ EPROM resident Upload/download utilities.

5.3 Binary vs ASCII Transfers

The basic memory unit of the 80386 microprocessor is the byte consisting of eight (8) binary bits, say b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0, in positions, 7,6, ... , 0 as shown in Figure 3. In this configuration, the most significant bit is b_7, and the least significant bit is b_0.

7	6	5	4	3	2	1	0
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Figure 3. Byte/Bit Configuration.

The meaning of any particular byte is of course a function of its use by some program or processor.

If it is desired to transmit the byte of Figure 3 exactly as it appears, this is said to be a binary transfer and the eight (8) data bits transmitted as a character by the SDK-386™ will be bits 7 through 0 of Figure 3.

In certain situations, as in this case, it may be desired to consider the bits 7 through 4 as a single unit, e.g., representing an ASCII HEXADECIMAL character of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Bits 3 through 0 may also be considered as representing a similar single unit or character. In fact, the most significant four bits and the least significant four bits are permitted to each represent a HEX character to be transmitted through the serial communication channel.

5.4 Using The SDK-386™ Upload Utility

Begin by pressing the [UPLD] key on the SDK-386™ keypad. The SDK-386™ will then display the message: **File----->**, with the last transfer mode entered displayed to the right of this prompt. At this point the user should either enter "0000000A" indicating an ASCII transfer is to be made or "0000000B" indicating a binary transfer is to be made. If the "0000000A" option is chosen, each byte is divided into two four bit segments. Each segment of each byte sent is then converted to it's corresponding ASCII code as described earlier before it is sent. For example, if the byte:

10011100

is to be sent to the computer connected to the SDK-386™ , a value 39 HEX will first be sent, since 39 HEX is the ASCII code for 9, the HEX value of the most significant four bit segment (nibble) of the above byte of data, followed by the value 43 HEX, which is the ASCII code for an upper case C, the HEX value of the least significant nibble of the above byte of data. In making this conversion, the code of the upper case letters (ASCII codes 41 HEX to 46 HEX) is sent for HEX values of A to F. Thus, for each byte of SDK-386™ memory to be transmitted, two 8 bit characters or bytes are transmitted. A complete list of the conversions made when the SDK-386™ is in the ASCII transfer mode is shown in Table 3.

If the "0000000B" option is selected, the byte is sent to the computer as is without any conversion. In the above example, the value 9C HEX would be sent.

After the "0000000A" or "0000000B" option has been entered (any other input will not be accepted), the user should press the "+" key. The SDK-386™ will then display the message: **Strt----->**, with the last starting address entered displayed at the right. At this point the user should enter the address of the first byte of the SDK-386™ RAM to be sent to the computer and press the "+" key. As with the "TAPW" routine, the program makes the standard bounds checks. If the "-" key is pressed, the **File----->** message is displayed again and the user can re-select the transfer mode option.

Next the SDK-386™ displays the message: **End ----->**, with the last ending address entered displayed at the right. At this point the user should enter the address of the last byte in SDK-386™ RAM to be sent to the computer and press the "+" key. As with the starting address, bounds checks are made. Also, if the starting address entered is larger than the ending address, an error message will be displayed. Pressing the "-" key will allow the user to re-enter the starting address.

After the ending address is entered, the "+" key pressed and the bounds checks are satisfied, the SDK-386™ begins sending the contents of the block of RAM specified to the computer, starting with the byte at the starting address and proceeding upward in memory, sending consecutive bytes of memory until the byte at the ending address has been sent. The SDK-386™ then displays the end address and it's contents in the data byte entry mode format last selected by the user and returns control to the monitor.

HEX CODE NUMBER	RAM REPRESENTATION 4 -BITS	ASCII REPRESENTATION (BYTE SENT) 8 BITS
0	0000	0011-0000
1	0001	0011-0001
2	0010	0011-0010
3	0011	0011-0011
4	0100	0011-0100
5	0101	0011-0101
6	0110	0011-0110
7	0111	0011-0111
8	1000	0011-1000
9	1001	0011-1001
A	1010	0100-0001
B	1011	0100-0010
C	1100	0100-0011
D	1101	0100-0100
E	1110	0100-0101
F	1111	0100-0110

Table 3.Ram Contents and Byte Sent for ASCII File Transfer

5.5 Using The SDK-386™ Download Utility

Begin by pressing the [UPLD] key on the SDK-386™'s keypad. The SDK-386™ will then display the message: **File----->**, with the last transfer mode entered displayed to the right of this prompt. At this point the user should either enter "0000000A" for ASCII or "0000000B" for binary.

If the "0000000A "option is chosen, each byte sent from the personal computer will be assumed to be in ASCII code and will be converted to it's binary equivalent, according to Table 3 before being stored in memory. Each byte send by the personal computer will therefore

correspond to a nibble of memory space in the SDK-386™ . Only bytes sent that correspond to the ASCII code for the digits 0 - 9 (30 HEX to 39 HEX) and the upper case letters A - F (41 HEX to 46 HEX) will be converted. All others will be ignored. This allows the user to type up an ASCII file continuing 68000 machine code on the personal computer using an editor or word processor program, inserting blanks and carriage returns to make it more readable, save it on floppy disk and send it to the SDK-386™ . Comments may even be inserted as long as lower case letters and no digits are used.

If the "0000000B" option is chosen, bytes sent from the computer are simply stored in memory as is with no conversion.

After the "0000000A" or "0000000B" option has been entered (no other input will be accepted), the user should press the "+" key.The SDK-386™ will then display the message: **Strt----->**, with the last starting address entered displayed at the right. At this point the user should enter the address in the SDK-386™ RAM where the first byte of the file is to be stored and press the "+" key. As before the program makes the standard bounds checks. If the "-" key is pressed, the **File----->** message is displayed again and the user can re-select the transfer mode option.

Next the SDK-386™ displays the message: **End ----->**, with the last ending address entered displayed at the right. At this point the user should enter the address in the SDK-386™ RAM where the last byte of the file is to be stored and press the "+" key. Again bounds checks are made. Also, if the starting address entered is larger than the ending address, an error message will be displayed. Pressing the "-" key will allow the user to re-enter the starting address.

After the end address is entered, the "+" key pressed and the bounds checks are satisfied, the SDK-386™ waits to receive the first byte of data from the computer. The user should then enter the necessary commands into the computer so that it will begin sending data to the SDK-386™ . As data are received from the computer, they are loaded into the SDK-386™ memory, beginning at the starting address and proceeding upward in memory, loading the in coming data into consecutive bytes of memory until the ending address is reached. The program then displays the end address and it's contents in the standard byte entry mode format and returns control to the monitor.

The SDK-386™ will wait indefinitely to receive the first byte of data. After the first byte has been received, the SDK-386™ will wait about one second for each successive byte of data. If another byte is not received after this amount of time, the SDK-386™ assumes the transmission is over and displays the last address in memory that data was loaded into and it's contents and returns control to the monitor. In this way, if the user is unsure about the size of the file to be sent to the SDK-386™ , an arbitrarily high address can be entered for the ending address and, after the computer has sent the last byte of data from the file, control is returned to the monitor even if the ending address has not been reached.

6.0 USING THE SDK-386™ WITH THE IBM-PC® AND AT®

This section describes the use of the PCOM-PC software package for the IBM-PC® and AT computers. When used with the EPROM resident monitor Upload/Download utilities described in Section 5.0, the PCOM-PC software completes the link between the SDK-386™ and the IBM-PC® or AT computer, thus allowing pre-assembled code generated by a cross assembler on the IBM-PC® to be transferred to the SDK-386™ . Additionally data and code may be stored on floppy disk and transferred between the SDK-386™ and the IBM-PC®.

The PCOM-PC software, supplied with the SDK-386™ on a 5 1/4" floppy disk, consists of two command files named DNLOAD.COM and UPLOAD.COM. DNLOAD.COM allows the user to transfer files stored on floppy disk from the PC® to the SDK-386™ through the COM1 port of the IBM-PC®, while UPLOAD.COM receives files from the SDK-386™ and stores them on floppy disk through the COM1 port of the IBM-PC®.

6.1 Transferring Files From The IBM PC® To The SDK-386™

First make all the required hardware connections between the SDK-386™ , and the IBM PC® as described in Section 5.1.1. All equipment should be turned off while making these connections. The 25DB connector on the SDK-386™ board should be connected to the COM1 port of the IBM PC®. Next, the SDK-386™ and the IBM PC® should be powered up. The DNLOAD.COM file should be present on the default drive and directory of the PC® (It is not absolutely necessary that DNLOAD.COM be available on the default drive and directory, but if it is not, it's path must be included when typing in the DNLOAD command).

To transfer a file from the PC® to the SDK-386™ , the user should begin by pressing the [DNLD] key on the SDK-386™ and starting it's download software utility, entering the correct transfer mode, starting address and ending address as described earlier in Section 5.5 . Next the user should type the command:

DNLOAD <pathname>[/<switch>]

into the IBM PC® where <pathname> is the path and name of the file to be transferred and <switch> is either s, a, or b specifying that an s-record, ASCII or binary type of file is to be sent to the SDK-386™ respectively. If no switch is specified, an s-record transfer is assumed by default. Each type of transfer is described in detail below.

6.1.1 S-Record Transfers The s-record transfer mode is used to transfer s-record ASCII files. Such files contain lines of ASCII characters called s-records which contain the 80386 machine code in HEX and the address in RAM of the first byte of code contained in that s-record, followed by several bytes of code. Refer to the user's manual of the particular assembler you are using to determine if it creates s-records files.

Since the s-record files are ASCII files, the user should select the transfer mode as "0000000A" when entering the information requested by the SDK-386™ download software.

As described previously, the SDK-386™ download software loads data into consecutive bytes of memory, beginning with the starting address until the ending address is reached. For this reason, the address field of consecutive s-records in the s-record file must be in ascending order. If the DNLOAD program encounters an out-of-order s-record address an error message is displayed on the screen. Error messages are also displayed if the file named in the command statement is not structured in the s-record format. If the DNLOAD program encounters an s-record containing an address higher than the address of the last byte of code in the previous s-record, the value FF HEX is sent to the SDK-386™ repeatedly until all memory between the address of the last byte of code sent and the next s-record is filled with FF HEX.

6.1.2 ASCII Transfers Typing the /a switch after the file name of a DNLOAD command specifies that an ASCII file is to be sent to the SDK-386™ . This transfer mode allows the user to create ASCII files containing 80386 machine code or data in hexadecimal on the IBM PC® and down load them to the SDK-386™ .

For this type of transfer the user should begin by starting up the SDK-386™ download software, entering the file type as "0000000A" and entering the desired starting and ending addresses as described in Section 5.5. Next, the user should type the command:

dnload filename.ext/a

into the IBM PC® (where filename is the name of the ASCII file to be transferred and ext is it's extension). The DNLOAD program will then begin sending ASCII characters to the SDK-386™ and displaying them on the IBM PC's® screen as they are sent. After the entire file has sent, the DNLOAD program returns back to DOS. As noted earlier the SDK-386™ , as it receives

characters, it will store their equivalent HEX values in memory. Only the characters 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F will be recognized by the SDK-386™ .

6.1.3 Binary Transfers Typing the /b switch after the file name of a DNLOAD command specifies that a data file is to be sent to the SDK-386™ . This transfer mode allows the user to transfer machine code data files to the SDK-386™ which were created by a cross assembler or other user application software on the IBM PC® or which were received from the SDK-386™ earlier, using the UPLOAD command (described later in Section 6.2).

For this type of transfer the user should begin by starting up the SDK-386™ download software, entering the file type as "0000000B" and entering the desired starting and ending addresses as described in Section 5.4. Next, the user should type the command:

dnload filename.ext/b

into the IBM PC® (where filename is the name of the file to be transferred and ext is it's extension). The DNLOAD program will then begin sending bytes of data to the SDK-386™ . No display is provided on the IBM PC® for this type of transfer. After the entire file has sent, the DNLOAD program returns back to DOS. As noted earlier the SDK-386™ stores the characters in memory as they are received without any interpretation for this type of transfer.

6.2 Transferring Files From The SDK-386™ To The IBM PC®

First the SDK-386™ should be connected to the COM1 port of the PC®, as described in Section 6.1 with the PC® and the SDK-386™ turned off. Next, the IBM-PC® and the SDK-386™ should be powered up and the UPLOAD.COM file should present on the default drive and directory of the PC® (As before it is not absolutely necessary that DNLOAD.COM be available on the default drive and directory, but if it is not, it's path must be included when typing in the UPLOAD command).

To transfer a file from the SDK-386™ to the IBM PC®, the user should begin by typing the command:

UPLOAD <pathname>[/<switch>]

into the IBM PC®, where <pathname> is the path and name of the file to be transferred and <switch> is either a, or b specifying that an ASCII or binary type of file is to be sent to the SDK-386™ respectively. If no switch is specified, ASCII transfer is assumed by default. Each type of transfer is described in detail below. Next the user press the [UPLD] key on the SDK-386™ to start up it's send file software, entering the correct file type, starting address and ending address as described in Section 5.4.

6.2.1 ASCII Transfers Typing the /a switch after the file name of a UPLOAD command specifies that an ASCII file is to be sent to the IBM PC®. This transfer mode allows the user to create and store ASCII files containing the hexadecimal data from a portion of the SDK-386™ memory. Such files can then be revised on the IBM PC® and down loaded back to the SDK-386™ using the DNLOAD command.

For this type of transfer the user should begin by typing the command:

upload filename.ext/a

into the IBM PC® (where filename is the name of the ASCII file to be transferred and ext is it's extension). Next the user should start up the SDK-386™ upload software, entering file type as "0000000A" and entering the desired starting address and ending address as described in Section 5.4. The UPLOAD program will then begin receiving ASCII characters from the SDK-386™, writing them to a file with the specified file name on disk and displaying them on the IBM PC's® screen as they are received. As noted earlier, in this mode of transfer, the SDK-386™

sends two ASCII characters, representing the HEX value contained in a byte of memory for each byte located between the starting address and the ending address. After the entire file has been sent, the UPLOAD program returns back to DOS. The file written to disk is formatted in bytes of data, with every two characters, representing one byte of the SDK-386™ RAM separated by a space, and each line containing 26 of these bytes.

6.2.2 Binary Transfers Typing the /b switch after the file name of an UPLOAD command specifies that a data file is to be sent from the SDK-386™ . This transfer mode allows the user to transfer data files to the IBM-PC® from the SDK-386™ were they can be modified by application software on the IBM PC® and, if desired, and sent back to the SDK-386™ using the DNLOAD command.

For this type of transfer, the user should begin by typing the command:

upload filename.ext/b

into the IBM PC® (where filename is the name of the ASCII file to be transferred and ext is it's extension). Next, the SDK-386™ upload software should be started , entering the file type as "0000000B" and entering the desired starting and ending addresses as described in Section 5.4. The UPLOAD program will then begin receiving data from the SDK-386™ and writing them to a file on floppy disk with the name specified. For this type of transfer, each byte of data received is equal to the contents of one byte of the SDK-386™ memory and the SDK-386™ sends one character for each byte located between the starting address and the ending address. No display is provided on the IBM PC® for this type of transfer. After the entire file has been sent, the UPLOAD program returns back to DOS.

7.0 PROGRAMMING CONSIDERATIONS

7.1 User Modifiable Exception Vectors

The SDK-386™ uses an Interrupt Descriptor Table which resides in RAM and extends from memory locations 000000A0H to 000001AFH. Exception gates are loaded into the interrupt table for exceptions 0 through 14 and exception 16 by the monitor as part of its initialization routine (see Section 4.1). These exceptions correspond to error conditions encountered by the processor, and the exception gates loaded into these locations of the Interrupt Descriptor Table call the monitor routines which display an appropriate error message. The exception handler routines for these exceptions may be redefined by the user by loading different exception gates into the Interrupt Descriptor Table. Care must be taken in loading alternate exception gates into these locations of the Interrupt Descriptor Table. However, because this may cause the monitor to operate incorrectly. Particular care must be taken if exception 2 (the non-maskable interrupt) is redefined, which is used by the [BREAK] key. Exceptions 32 and 33 are undefined and are provided for use by the user (a level 32 interrupt is generated by the [USER INT] key, see Section 4.5.1). The user should consult the 80386 Programer's Reference Manual to determining the exact format of the desired exception gate. The memory locations of each exception gate in the Interrupt Descriptor Table are listed in Table 4.

Exception No.	RAM Address	Exception
0	000000A0H	Zero Divide
1	000000A8H	Debug Exception
2	000000B0H	Non Maskable Interrupt
3	000000B8H	Break Point Exception
4	000000C0H	Overflow Exception
5	000000C8H	Bounds Check
6	000000D0H	Invalid Opcode
7	000000D8H	Coprocessor Not Available

8	000000E0H	Double Fault
9	000000E8H	Coprocessor Seg. Overrun
10	000000F0H	Invalid TSS
11	000000F8H	Seg. Not Present
12	00000100H	Stack Exception
13	00000108H	Gen. Protection Error
14	00000110H	Page Fault
15	00000118H	Reserved
16	00000120H	Coprocessor Error
17	00000128H	Reserved
18	00000130H	Reserved
19	00000138H	Reserved
20	00000140H	Reserved
21	00000148H	Reserved
22	00000150H	Reserved
23	00000158H	Reserved
24	00000160H	Reserved
25	00000168H	Reserved
26	00000170H	Reserved
27	00000178H	Reserved
28	00000180H	Reserved
29	00000188H	Reserved
30	00000190H	Reserved
31	00000198H	Reserved
32	000001A0H	User Int Key
33	000001A8H	User Definable

Table 4. Interrupt Descriptor Table.

7.2 Utilization of Monitor Subroutines

User routines can call the monitors subroutines where required. This section provides the user with information concerning several of the routines for which the user may have need.

7.2.1 Keyboard Routine The keyboard routine (located at address 00FF9874H) displays the seventeen character message in display buffer (beginning at 000002ECH) and scans the keyboard for a key closure. Instruction execution will remain in this routine until a key closure is detected. This is accomplished by continually scanning until a key closure is detected. The key code for the closure is returned in AL. Table 5 lists the key code values associated with the different keys on the keypad. All of the registers are saved except for EAX and EBX. The BL register determines the interpretation of the **Shift** key. If BL is equal to 00H and the shift key is pressed, a value of 10H is returned in AL as the key code. If BL is not equal to 00H shift key will not cause a return from the keyboard routine. The display is then changed to that of the **Shift** message. A second closure of the shift key clears the display. The shift key is used internally to the keyboard routine to shift the key code table base to the second function of the keys and back again to the bottom of the table. Upon entry to the keyboard routine, it is important that BL be loaded correctly, according to the desired action in response to **Shift** key press.

7.2.2 The Display Routine The Display routine (located at address 00FF9767H) writes a 17 character message in display buffer (beginning at 000002ECH) to the first line of the LCD display. This routine should be used instead of the Keyboard routine in Section 7.3.1 when it is not required to change the message in the LCD display, and a key press from the user is not needed. The hexadecimal codes for the alpha - numeric characters are given in Table 6. Messages should be placed into memory with the left most character in the lowest memory location and the right

most character in the highest. Once written to the LCD display a message will continue to be displayed until a new message is written to the display. Therefore, a message of 17 spaces (HEX code 20H) must be written to the display to clear it.

First Key	Function Code	Second Key	Function Code	First Key	Function Code	Second Key	Function Code
0	00H	none	30H	UPLOAD	14H	TPWT	24H
1	01H	none	31H	STEP	15H	none	25H
2	02H	none	32H	DWD	16H	none	26H
3	03H	none	33H	+ (plus)	17H	none	27H
4	04H	none	34H	DNLD	18H	TPRD	28H
5	05H	none	35H	INS	19H	DEL	29H
6	06H	none	36H	WORD	1AH	none	2AH
7	07H	none	37H	R32	1BH	R16	2BH
8	08H	none	38H	MOV	1CH	none	2CH
9	09H	none	39H	AUTO	1DH	none	2DH
A	0AH	none	3AH	BYTE	1EH	none	2EH
B	0BH	none	3BH	ADDR	1FH	REL	2FH
C	0CH	none	3CH	U0	40H	SU0	50H
D	0DH	none	3DH	U1	41H	SU1	51H
E	0EH	none	3EH	U2	42H	SU2	52H
F	0FH	none	3FH	U3	43H	SU3	53H
SHIFT	10H	none	20H	U4	44H	SU4	54H
RUN	11H	none	21H	U5	45H	SU5	55H
BRPT	12H	none	22H	U6	46H	SU6	56H
- (minus)	13H	none	23H	U7	47H	SU7	57H

Table 5. Key Hex Codes

CHAR	CODE	CHAR	CODE	CHAR	CODE	CHAR	CODE
A	41H	a	61H	N	4EH	n	6EH
B	42H	b	62H	O	4FH	o	6FH
C	43H	c	63H	P	50H	p	70H
D	44H	d	64H	Q	51H	q	71H
E	45H	e	65H	R	52H	r	72H
F	46H	f	66H	S	53H	s	73H
G	47H	g	67H	T	54H	t	74H
H	48H	h	68H	U	55H	u	75H
I	49H	i	69H	V	56H	v	76H
J	4AH	j	6AH	W	57H	w	77H
K	4BH	k	6BH	X	58H	x	78H
L	4CH	l	6CH	Y	59H	y	79H
M	4DH	m	6DH	Z	5AH	z	7AH
0	30H	3	33H	6	36H	9	39H
1	31H	4	34H	7	37H	(space)	20H
2	32H	5	35H	8	38H	- (dash)	B0H

Table 6. Display Characters and Their Hex Codes

7.2.3 The Format Character Routine The Format Character routine (located at address 00FF99E8H) generates the character codes for the hexadecimal digits located at the address pointed to by register ESI. The number of bytes formatted is equal to the value of register ECX.

Thus, consecutive bytes of data are read, starting with the byte pointed to by ESI, formatted, and their character codes are loaded into consecutive memory locations, the first character code being loaded into the memory location pointed to by EDI. To use the Format Character routine, ESI should be loaded with the address of the first byte of data to be formatted, EDI should be loaded with the address of the leftmost digit in the display buffer of where the data are to appear on the display, and ECX should be loaded with the number of bytes to be formatted. Each byte of data formatted results in two character codes, thus a total of 8 bytes, resulting in 16 character codes can be formatted and displayed in the 17 digit format of the SDK-386™ display.

7.2.4 The Tone Routine The Tone Routine (located at address 00FF985BH) generates a square wave which is sent to the speaker located on the SDK-386™ board. Upon entry, ECX contains the number which is to be proportional to one half the period of the square wave and EBX contains the number of periods to be generated. An approximate calculation of the period when a 10 megahertz clock is used is as follows:

Period = 13+ 74.5* (ECX)
= the period in μsec.

8.0 Example of Programming/Execution

The following is an example of loading and executing a program with the SDK-386™.

The assembly/machine language program is as follows:

LOCATION	CONTENTS	LABEL	INSTRUCTION
00000400	90	START:	NOP
00000401	90		NOP
00000402	EB FC		JMP START

To load the program, press the following key:

ADDR

The display should read:

00000300 <-xxxxxx UV

where UV represents the contents of the byte at the given address (00000300H).

Now press the following sequence:

0 ; 0 ; 0 ; 0 ; 0 ; 4 ; 0 ; 0 ; BYTE

The display should read:

00000400 ->xxxxxx UV

where UV represents the contents of memory location 0400H.

Now press **9 ; 0** resulting in the display

0000040->xxxxxx90

Now press **+** giving the display

00000401->xxxxxxUV

Again, UV is whatever is contained in location 0401H.

Now press **9 ; 0** resulting in the display

00000401->xxxxxx90

Now press **+** giving the display

00000402->xxxxxxUV

Now press **E ; B** resulting in the display

00000402->xxxxxxEB

Now press **+** giving the display

00000403->xxxxxxUV

Now press **F ; C** resulting in the display

00000403->xxxxxxFC

The example program is now loaded. To execute the program, the Extended Instruction Pointer (EIP) must be set to 00000400H. To do this, press the key **R32**. The display should read:

CR3-----<-00000000

Now press the **EIP** key giving the display

EIP-----<-00000300

Now press the **INS** (to **INS**ert data) key giving the following display

EIP------>00000300

Now press the key sequence (note the display changes with each key pressed).

0 ; 0 ; 0 ; 0 ; 0 ; 4 ; 0 ; 0

The display should read

EIP------>00000400

The EIP register is now initialized. To exit the display routine, press **+** giving the display

EIP-----<-00000400

Now press **+** one more time to return to the Address/Content format display.

00000403<-xxxxxxFC

To begin execution, press **RUN**. The display should go blank. The program is executing.

To provide an orderly stopping of the program, press the **BREAK** button. The address on the display is the address of the next instruction to be executed, e.g.,

00000401<-xxxxxx90

indicates the NOP instruction at location 0401H is the next instruction to be executed.

To execute the program one instruction at a time, now press the **STEP** key a number of times.

To initiate execution again, press the **RUN** key.

The above procedure can be repeated with any user program by making appropriate changes to perform what the user wants the program to do.

BLANK PAGE

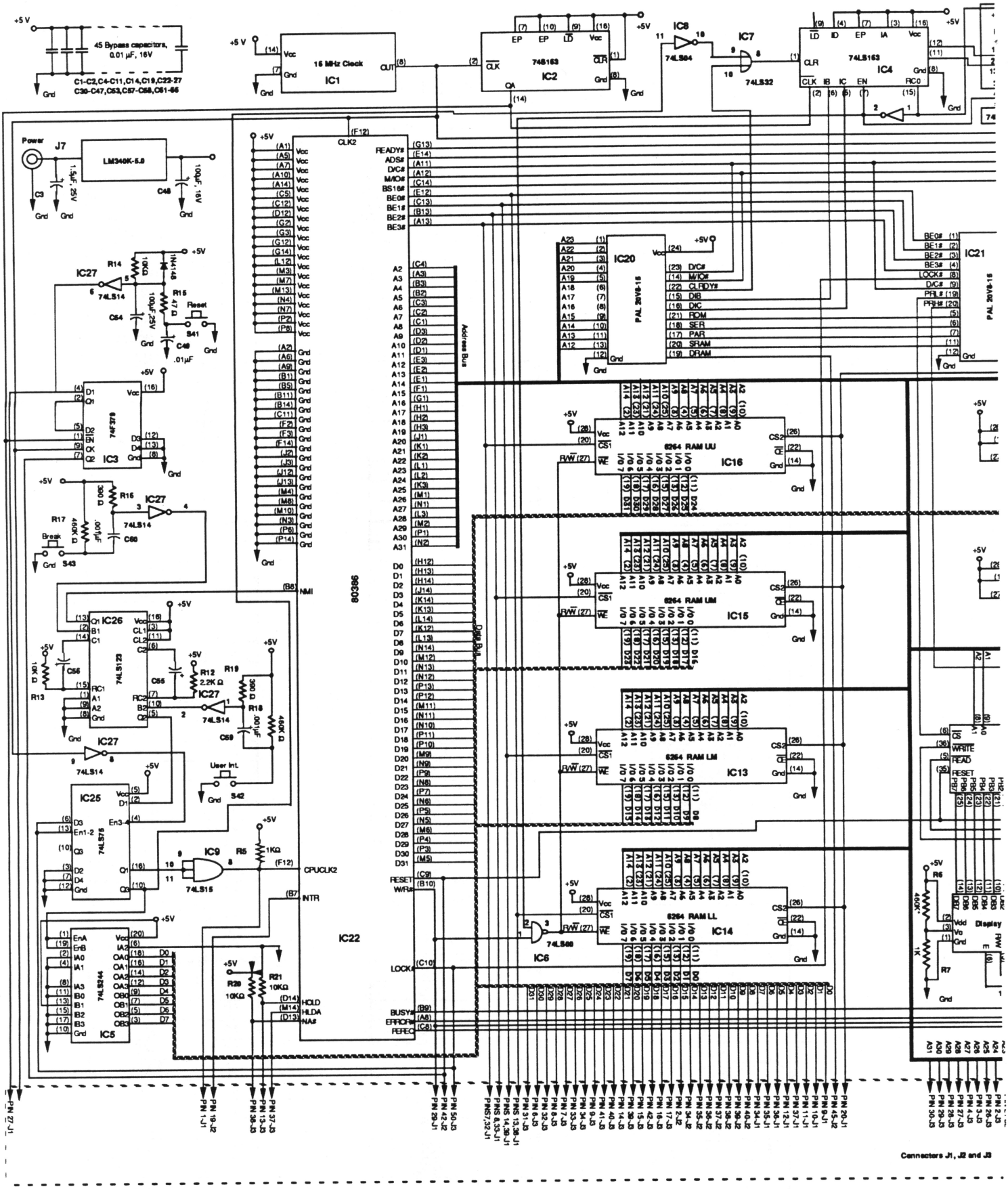
9.0 PINOUTS AND SCHEMATIC

Note: NC implies No Connection

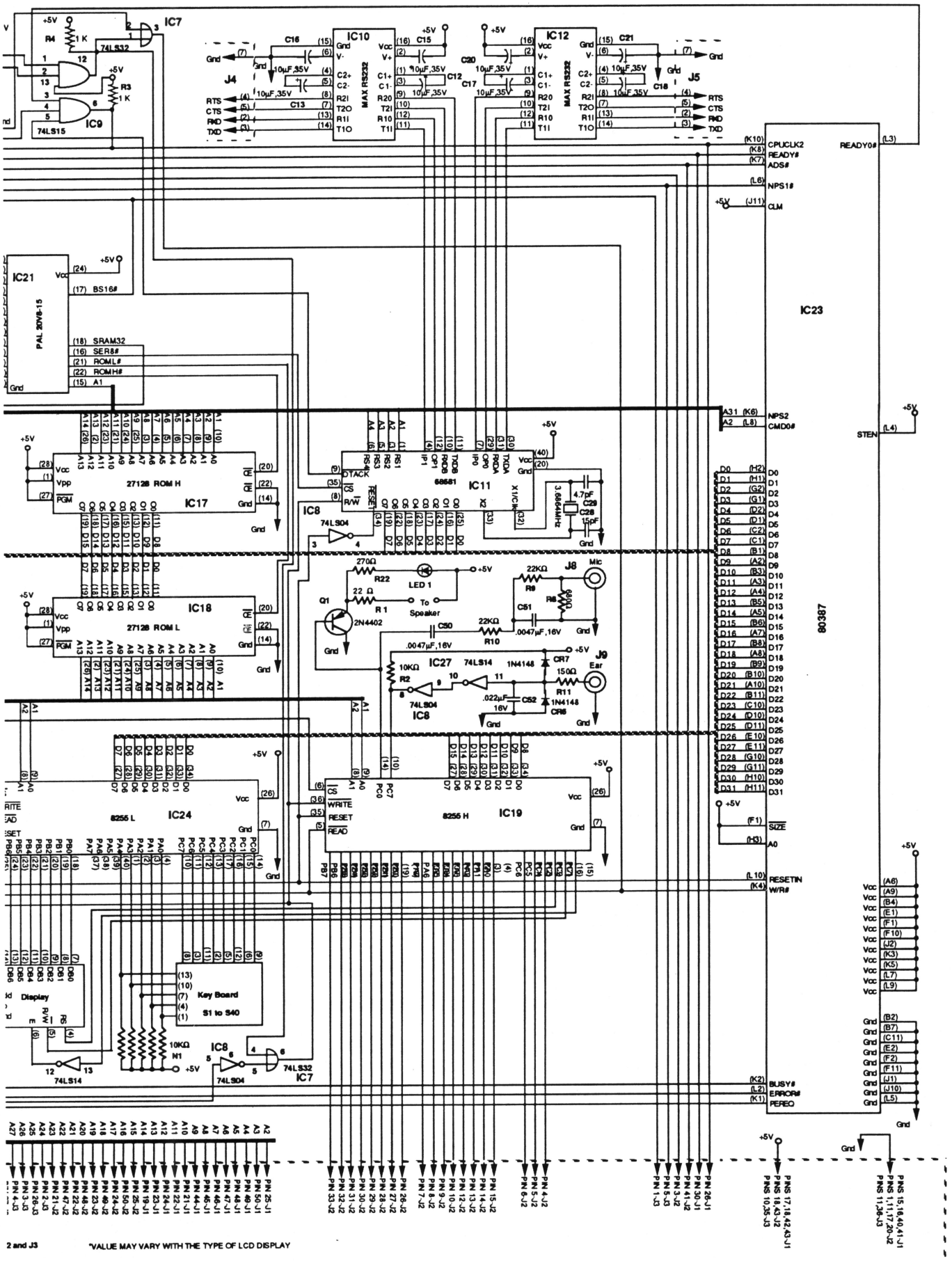
Connector J1	
Pin	Signal
1	CPUCLK2
2	NC
3	NC
4	NC
5	NC
6	NC
7	CS1UU
8	CS1UM
9	D0
10	D1
11	D2
12	D4
13	CS1LL
14	CS1LM
15	GND
16	GND
17	+5B
18	+5V
19	A14
20	CS2
21	A10
22	A11
23	A13
24	A12
25	A2
26	CPUCLK2
27	not(CLOCK)
28	NC
29	W/not(R)
30	not(READY)
31	NC
32	CS1UU
33	CS1UM
34	D7
35	D6
36	D5
37	D3
38	CS1LL
39	CS1LM
40	GND
41	GND
42	+5V
43	+5V
44	A9
45	A8
46	A7
47	A6
48	A5
49	A4
50	A3

Connector J2	
Pin	Signal
1	GND
2	D15
3	D/not(C)
4	PC4H
5	PC5H
6	PC6H
7	PA7H
8	PA6H
9	PA5H
10	PA4H
11	GND
12	PA3H
13	PA2H
14	PA1H
15	PA0H
16	RESET
17	GND
18	+5V
19	INTR
20	GND
21	A23
22	A21
23	A19
24	A17
25	A15
26	PB0H
27	PB1H
28	PB2H
29	PB3H
30	PB4H
31	PB5H
32	PB6H
33	PB7H
34	D14
35	D13
36	D12
37	D11
38	D10
39	D9
40	D8
41	not(ADS)
42	RESET
43	+5V
44	NC
45	DRAM
46	NC
47	A22
48	A20
49	A18
50	A16

Connector J3	
Pin	Signal
1	not(BS16)
2	A24
3	A26
4	A27
5	not(NPS1)
6	D30
7	D27
8	D28
9	D24
10	+5V
11	GND
12	NC
13	HOLD
14	D21
15	D19
16	D17
17	D16
18	PA6L
19	NC
20	NC
21	NC
22	NC
23	NC
24	NC
25	NC
26	A25
27	A28
28	A29
29	A30
30	A31
31	D31
32	D29
33	D26
34	D25
35	+5V
36	GND
37	HLDA
38	not(NA)
39	D20
40	D22
41	D23
42	D18
43	PA7L
44	PA5L
45	NC
46	NC
47	NC
48	NC
49	NC
50	not(LOCK)



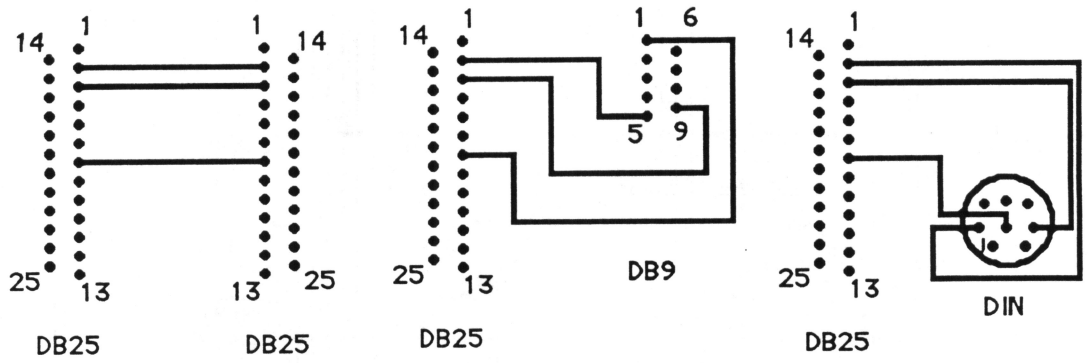
NOTES:(1) Chips with pin numbers in bold face type indicates chip appears as logical function parts as opposed to one block on the schematic.



10.0 CABLING

The SDK-386™ has two (2) female DB25 connectors, one for Port A (connector J5) and the other for Port B (connector J4). The software is written for channel A although either channel may be used.

The cable connections required for either channel A or channel B are shown in Figure 4.



a. Cable Connections to the IBM PC®.
b. Cable Connections to the Macintosh® 512K.
c. Cable connections to the Macintosh® Plus.

Figure 4. SDK-386™ Cable Connections

11.0 REFERENCES

Although some background in microprocessor theory and hardware implementation is necessary to be able to use this SDK-386™ User's Manual effectively on an individual basis, the optimal use of the SDK-386™ is probably in an instructional course on microprocessors where the general concepts are illustrated by using the SDK-386™. Nevertheless, listed below are several references which you may find useful.

1. Programming the 8086 Microprocessor Using the P8086 µLAB™, Marlin H. Mickle and William G. Vogt, Unpublished Manuscript, 1988

2. Computer Organization and Architecture, William Stallings, Macmillan, New Jersey, 1986.

3. Microprocessor-Based System Design, D. J. Comer, Holt, Rinehart and Winston, New York, 1986.

4. Microcomputer Experimentation with the INTEL® SDK-86, Lance A. Leventhal, Holt, Rinehart and Winston, New York, 1987.

5. COMPLETE GUIDE TO RS-232 AND PARALLEL CONNECTIONS: A Step-by-Step Approach to Connecting Computers, Printers, Terminals, and Modems, Martin D. Seyer, Prentice Hall, Englewood Cliffs, NJ, 1988.

6. Microprocessors and Interfacing, Programming and Hardware, Douglas V. Hall, Mc-Graw-Hill, 1986. (See Intel Literature Guide), ISBN 0-07-025526-1

7. The 8086 and 80286 Microprocessors, Hardware and Software Interfacing, Avtar Singh and Walter A. Triebel, Prentice Hall, 1990, ISBN 0-13-245325-8.

8. 80386 Assembly Language : A Complete Tutorial and Subroutine Library, Penn Brumm and Don Brumm. -- 1st ed. -- Blue Ridge Summit, PA : Tab Professional and Reference Books, 1988.

9. The 80386/387 Architecture Stephen P. Morse, Eric J. Isaacson, Douglas J. Albert. -- New York : Wiley, 1987.

10. 80386 Microprocessor Handbook Chris H. Pappas and William H. Murray III. -- Berkeley, Calif. : Osborne McGraw-Hill, 1988.

11. 80286 and 80386 Microprocessors : New PC Architectures, A.B. Fontaine and F. Barrand ; translated by A. Rawsthorne. -- New York : Van Nostrand Reinhold, 1989.

12. The Intel Microprocessors: 8086/8088, 80186, 80286, 80386, and 80486: Architecture, Programming, and Interfacing,.Barry B. Brey. -- 2nd ed. -- New York : Merrill, 1991.

13. Intel's Official Guide to 386 Computing, Michael Edelhart. -- Berkeley, Calif. : Osborne McGraw-Hill, 1991.

14. 80386 Hardware Reference Manual, Intel -- Santa Clara, CA: Intel Corporation, 1987

15. 386 Microprocessor Hardware Reference Manual, Intel -- Santa Clara, CA: Intel Corporation, 1988.

16. 80386 System Software Writer's Guide, Intel -- Santa Clara, CA: Intel Corporation, 1987.

17. 80386 programmer's reference manual, Intel -- Santa Clara, CA: Intel Corporation, 1986.

18. 386™ SX Microprocessor Programmer's Reference Manual, Intel ®, Osborne, McGraw-Hill,1989.

19. 386™ DX Microprocessor Programmer's Reference Manual, 1990, Intel -- Santa Clara, CA: Intel Corporation, 1989

BLANK PAGE

12.0 OBJECT AND ASSEMBLY CODE LISTINGS OF THE SDK-386™

```
2500 A.D. 80386 Cross Assembler  -  Version 4.03c
-------------------------------------------------

        Input  Filename : SDK386.asm
        Output Filename : SDK386.obj


1                                       ;SYSTEM DATA STORAGE LOCATIONS
2   0000:00000000                               .DATA
3   0000:00000000
4                                   ;---------------------------------------------------------------------
5                                       ;GLOBAL DESCRIPTOR TABLE
6                                   ;---------------------------------------------------------------------
7   0000:00000000                               ORG 0000H
8                                       NULL_G_DESC:                                    ;NULL
9   0000:00000008                               ORG 0008H
10                                      CODE_G_DESC:                                    ;CODE
11  0000:00000010                               ORG 0010H
12                                      DATA_G_DESC:                                    ;DATA
13  0000:00000018                               ORG 0018H
14  0000:00000018               LDT_G_DESC:
15  0000:00000020                               ORG 0020H
16                                      SYS_TASK_G_DESC:                                ;SYSTEM TASK
17  0000:00000028                               ORG 0028H
18                                      USER_TASK_G_DESC:                               ;USER TASK SEGMENT DESCRIPTOR
19
20                                  ;---------------------------------------------------------------------
21                                  ;LOCAL DESCRIPTOR TABLE
22                                  ;---------------------------------------------------------------------
23  0000:00000030                               ORG 0030H
24                                      NULL_L_DESC:                                    ;NULL DESCRIPTOR
25  0000:00000038                               ORG 0038H
26                                      CODE_L_DESC:                                    ;CODE DESCRIPTOR
27  0000:00000040                               ORG 0040H
28                                      DATA_L_DESC:                                   ;DATA SEGMENT DESCRIPTOR
29  0000:00000048                               ORG 0048H
30                                      USER_L_DESC1:                                  ;USER DEFINED DESCRIPTOR
31  0000:00000050                               ORG 0050H
32                                      USER_L_DESC2:                                  ;USER DEFINED DESCRIPTOR
33  0000:00000058                               ORG 0058H
34                                      USER_L_DESC3:                                  ;USER DEFINED DESCRIPTOR
35
36                                  ;---------------------------------------------------------------------
37                                   ;INTERRUPT DESCRIPTOR TABLE
38                                  ;---------------------------------------------------------------------
39  0000:00000060                               ORG 0060H
40                                      DIVIDE_ERROR:                                   ;0 DIVIDE EXCEPTION
41  0000:00000068                               ORG 0068H
42                                      DEBUG_EXCEP:                                    ;DEBUG EXCEPTION
43  0000:00000070                               ORG 0070H
44                                      NMI:                                            ;BREAK KEY
45  0000:00000078                               ORG 0078H
46                                      BREAK_PT;                                       ;BEAKPOINT EXCEPTION
47  0000:00000080                               ORG 0080H
48                                      OVERFLOW:                                       ;OVERFLOW EXCEPTION
49  0000:00000088                               ORG 0088H
50                                      BOUNDS_ERROR:                                  ;BOUNDS ERROR EXCEPTION
51  0000:00000090                               ORG 0090H
52                                      INVALID_OPCODE:                                 ;OPCODE ERROR
53  0000:00000098                               ORG 0098H
54                                      COPRO_NOT_AVAL:                               ;COPROCESSOR NOT AVAILABLE
55
56  0000:0000009C                         ORG 009CH
57  0000:0000009C               MEM_VAL:
58
59  0000:000000A0                               ORG 00A0H
60                                      DOUBLE_FAULT:                                  ;DOUBLE FAULT EXCEPTION
61  0000:000000A8                               ORG 00A8H
62                                      COPRO_OVERRUN:                               ;COPROCESSOR SEGMENT OVERRUN
63  0000:000000B0                               ORG 00B0H
64                                      INVALID_TSS:                                    ;TSS INVALID
65  0000:000000B4                         ORG 00B4H
66  0000:000000B4               USE_EFLAGS:
67  0000:000000B8                               ORG 00B8H
68                                      SEG_NOT_PRES:                                   ;SEGMENT NOT PRESENT
69  0000:000000C0                               ORG 00C0H
70                                      STACK_EXCEP:                                    ;STACK EXCEPTION
71  0000:000000C8                               ORG 00C8H
72                                      GENERAL_PROTECTION:                           ;GENERAL PROTECTION ERROR
73  0000:000000D0                               ORG 00D0H
74                                      PAGE_FAULT:                                     ;PAGE FAULT EXCEPTION
75  0000:000000D0
76                                      ;-----------INTERRUPT # 15 RESERVED BY INTEL --------------------
77  0000:000000D0
78  0000:000000E0                               ORG 00E0H
79                                      COPRO_ERROR:                                    ;COPROCESSOR ERROR
80  0000:000000E0
81                                      ;------------INTERRUPT # 17-31 RESERVED BY INTEL-----------------
82  0000:000000E0
83  0000:00000160                               ORG 0160H
84                                      USER_INT_KEY:                                   ;USER INTERRUPT KEY
85  0000:00000168                               ORG 0168H
86                                      USER_DEFINEABLE:                ;USER DEFINEABLE INTERRUPT DESCRIPTOR
```



```
0000:00000168
0000:00000168
0000:00000168

                        ;-------------------------------------------------------------------------
                        ;SYSTEM TASK STATE SEGMENT
                        ;-------------------------------------------------------------------------
0000:000001B0                           ORG 01B0H
0000:000001B0           SYS_BACKLINK:
0000:000001B4                           ORG 01B4H
0000:000001B4           SYS_ESP0:
0000:000001B8                           ORG 01B8H
0000:000001B8           SYS_SS0:
0000:000001BC                           ORG 01BCH
0000:000001BC           SYS_ESP1:
0000:000001C0                           ORG 01C0H
0000:000001C0           SYS_SS1:
0000:000001C4                           ORG 01C4H
0000:000001C4           SYS_ESP2:
0000:000001C8                           ORG 01C8H
0000:000001C8           SYS_SS2:
0000:000001CC                           ORG 01CCH
0000:000001CC           SYS_CR3:
0000:000001D0                           ORG 01D0H
0000:000001D0           SYS_EIP:
0000:000002D4                           ORG 02D4H
0000:000002D4           SYS_EFLAGS:
0000:000001D8                           ORG 01D8H
0000:000001D8           SYS_EAX:
0000:000001DC                           ORG 01DCH
0000:000001DC           SYS_ECX:
0000:000001E0                           ORG 01E0H
0000:000001E0           SYS_EDX:
0000:000001E4                           ORG 01E4H
0000:000001E4           SYS_EBX:
0000:000001E8                           ORG 01E8H
0000:000001E8           SYS_ESP:
0000:000001EC                           ORG 01ECH
0000:000001EC           SYS_EBP:
0000:000001F0                           ORG 01F0H
0000:000001F0           SYS_ESI:
0000:000001F4                           ORG 01F4H
0000:000001F4           SYS_EDI:
0000:000001F8                           ORG 01F8H
0000:000001F8           SYS_ES:
0000:000001FC                           ORG 01FCH
0000:000001FC           SYS_CS:
0000:00000200                           ORG 0200H
0000:00000200           SYS_SS:
0000:00000204                           ORG 0204H
0000:00000204           SYS_DS:
0000:00000208                           ORG 0208H
0000:00000208           SYS_FS:
0000:0000020C                           ORG 020CH
0000:0000020C           SYS_GS:
0000:00000210                           ORG 0210H
0000:00000210           SYS_LDT:
0000:00000216                           ORG 0216H
0000:00000216           SYS_IO_MAP:
0000:00000216

                        ;-------------------------------------------------------------------------
                        ;SYSTEM  DATA STORAGE
                        ;-------------------------------------------------------------------------

0000:00000218                           ORG 0218H
0000:00000218           FREQ:
0000:0000021A                           ORG 021AH
0000:0000021A           DURATION:

0000:0000021C             ORG 021CH
0000:0000021C           A_MODE:


                        ;-------------------------------------------------------------------------
                        ;USER TASK STATE SEGMENT
                        ;-------------------------------------------------------------------------

0000:00000220                           ORG 0220H
0000:00000220           USER_BACKLINK:
0000:00000224                           ORG 0224H
0000:00000224           USER_ESP0:
0000:00000228                           ORG 0228H
0000:00000228           USER_SS0:
0000:0000022C                           ORG 022CH
0000:0000022C           USER_ESP1:
0000:00000230                           ORG 0230H
0000:00000230           USER_SS1:
0000:00000234                           ORG 0234H
0000:00000234           USER_ESP2:
0000:00000238                           ORG 0238H
0000:00000238           USER_SS2:
0000:0000023C                           ORG 023CH
0000:0000023C           USER_CR3:
0000:00000240                           ORG 0240H
0000:00000240           USER_EIP:
0000:00000244                           ORG 0244H
0000:00000244           USER_EFLAGS:
0000:00000248                           ORG 0248H
0000:00000248           USER_EAX:
0000:0000024C                           ORG 024CH
0000:0000024C           USER_ECX:
```

```
0000:00000250                                   ORG 0250H
0000:00000250                           USER_EDX:
0000:00000254                                   ORG 0254H
0000:00000254                           USER_EBX:
0000:00000258                                   ORG 0258H
0000:00000258                           USER_ESP:
0000:0000025C                                   ORG 025CH
0000:0000025C                           USER_EBP:
0000:00000260                                   ORG 0260H
0000:00000260                           USER_ESI:
0000:00000264                                   ORG 0264H
0000:00000264                           USER_EDI:
0000:00000268                                   ORG 0268H
0000:00000268                           USER_ES:
0000:0000026C                                   ORG 026CH
0000:0000026C                           USER_CS:
0000:00000270                                   ORG 0270H
0000:00000270                           USER_SS:
0000:00000274                                   ORG 0274H
0000:00000274                           USER_DS:
0000:00000278                                   ORG 0278H
0000:00000278                           USER_FS:
0000:0000027C                                   ORG 027CH
0000:0000027C                           USER_GS:
0000:00000280                                   ORG 0280H
0000:00000280                           USER_LDT:
0000:00000286                                   ORG 0286H
0000:00000286                           USER_IO_MAP:
0000:00000288                                   ORG 0288H
0000:00000288                           GDT_PTR:
0000:00000290                                   ORG 0290H
0000:00000290                           IDT_PTR:
0000:00000298                                   ORG 0298H
0000:00000298                           LAST_KEY:
0000:0000029C                                   ORG 029CH
0000:0000029C                           SYS_FLAGS:
0000:000002A0                                   ORG 02A0H
0000:000002A0                           CURRENT_ADDRESS:
0000:000002A4                                   ORG 02A4H
0000:000002A4                           DATA_MODE:
0000:000002A8                                   ORG 02A8H
0000:000002A8                           START_ADDRESS:
0000:000002AC                                   ORG 02ACH
0000:000002AC                           END_ADDRESS:
0000:000002B0                                   ORG 02B0H
0000:000002B0                           DEST_ADDRESS:
0000:000002B4                                   ORG 02B4H
0000:000002B4                           DEBUG_REG_0:
0000:000002B8                                   ORG 02B8H
0000:000002B8                           DEBUG_REG_1:
0000:000002BC                                   ORG 02BCH
0000:000002BC                           DEBUG_REG_2:
0000:000002C0                                   ORG 02C0H
0000:000002C0                           DEBUG_REG_3:
0000:000002C4                                   ORG 02C4H
0000:000002C4                           DEBUG_REG_7:

0000:000002C8                             ORG 02C8H
0000:000002C8                   MEM_VAL_3:

0000:000002CA                                   ORG 02CAH
0000:000002CA                           POWER_UP:

0000:000002CC                             ORG 02CCH
0000:000002CC                   B_DATA:

0000:000002B0                                   ORG 02B0H
0000:000002B0                           TEMP_0:

0000:000002D0                             ORG 02D0H
0000:000002D0                   I_DATA:
0000:000002D4                             ORG 02D4H
0000:000002D4                   H_DATA:
0000:000002D8                             ORG 02D8H
0000:000002D8                   MEM_VAL_2:
0000:000002DC                             ORG 02DCH
0000:000002DC                   G_DATA:
0000:000002E0                             ORG 02E0H
0000:000002E0                   F_DATA:
0000:000002E4                             ORG 02E4H
0000:000002E4                   E_DATA:
0000:000002E8                             ORG 02E8H
0000:000002E8                   MOVE_INFO:

0000:000002EC                                 ORG 02ECH
0000:000002EC                           DISPBF:
0000:00000000                                 .CODE
0000:00FF8000                                 ORG 00FF8000H
0000:00FF8000  90                             NOP
0000:00FF8001  90                             NOP
0000:00FF8002  90                             NOP
0000:00FF8003  90                             NOP
0000:00FF8004  90                       INIT: NOP
0000:00FF8005  90                             NOP
0000:00FF8006  90                             NOP
0000:00FF8007  90                             NOP
0000:00FF8008  FA                             CLI                     ;DISABLE INTERRUPTS
```

```
                                        ;SET UP GDT
0000:00FF8009   C6 06 00 00 00                          MOV NULL_G_DESC,BYTE PTR 00H    ;LOAD NULL DESCRIPTOR
0000:00FF800E   C6 06 01 00 00                          MOV NULL_G_DESC+1,BYTE PTR 00H
0000:00FF8013   C6 06 02 00 00                          MOV NULL_G_DESC+2,BYTE PTR 00H
0000:00FF8018   C6 06 03 00 00                          MOV NULL_G_DESC+3,BYTE PTR 00H
0000:00FF801D   C6 06 04 00 00                          MOV NULL_G_DESC+4,BYTE PTR 00H
0000:00FF8022   C6 06 05 00 00                          MOV NULL_G_DESC+5,BYTE PTR 00H
0000:00FF8027   C6 06 06 00 00                          MOV NULL_G_DESC+6,BYTE PTR 00H
0000:00FF802C   C6 06 08 00 FF                          MOV CODE_G_DESC,BYTE PTR FFH    ;LOAD CODE DESCRIPTOR
0000:00FF8031   C6 06 09 00 FF                          MOV CODE_G_DESC+1,BYTE PTR FFH
0000:00FF8036   C6 06 0A 00 00                          MOV CODE_G_DESC+2,BYTE PTR 00H
0000:00FF803B   C6 06 0B 00 00                          MOV CODE_G_DESC+3,BYTE PTR 00H
0000:00FF8040   C6 06 0C 00 00                          MOV CODE_G_DESC+4,BYTE PTR 00H
0000:00FF8045   C6 06 0D 00 9F                          MOV CODE_G_DESC+5,BYTE PTR 9FH
0000:00FF804A   C6 06 0E 00 CF                          MOV CODE_G_DESC+6,BYTE PTR CFH
0000:00FF804F   C6 06 0F 00 00                          MOV CODE_G_DESC+7,BYTE PTR 00H
0000:00FF8054   C6 06 10 00 FF                          MOV DATA_G_DESC,BYTE PTR FFH    ;LOAD DATA DESCRIPTOR
0000:00FF8059   C6 06 11 00 FF                          MOV DATA_G_DESC+1,BYTE PTR FFH
0000:00FF805E   C6 06 12 00 00                          MOV DATA_G_DESC+2,BYTE PTR 00H
0000:00FF8063   C6 06 13 00 00                          MOV DATA_G_DESC+3,BYTE PTR 00H
0000:00FF8068   C6 06 14 00 00                          MOV DATA_G_DESC+4,BYTE PTR 00H
0000:00FF806D   C6 06 15 00 93                          MOV DATA_G_DESC+5,BYTE PTR 93H
0000:00FF8072   C6 06 16 00 CF                          MOV DATA_G_DESC+6,BYTE PTR CFH
0000:00FF8077   C6 06 17 00 00                          MOV DATA_G_DESC+7,BYTE PTR 00H
0000:00FF807C   C6 06 88 02 4F                          MOV GDT_PTR,BYTE PTR 4FH        ;LOAD GDT POINTER
0000:00FF8081   C6 06 89 02 00                          MOV GDT_PTR+1,BYTE PTR 00H
0000:00FF8086   C6 06 8A 02 00                          MOV GDT_PTR+2,BYTE PTR 00H
0000:00FF808B   C6 06 8B 02 00                          MOV GDT_PTR+3,BYTE PTR 00H
0000:00FF8090   C6 06 8C 02 00                          MOV GDT_PTR+4,BYTE PTR 00H
0000:00FF8095   C6 06 8D 02 00                          MOV GDT_PTR+5,BYTE PTR 00H
0000:00FF809A   C6 06 90 02 0F                          MOV IDT_PTR,BYTE PTR 0FH        ;LOAD IDT POINTER
0000:00FF809F   C6 06 91 02 01                          MOV IDT_PTR+1,BYTE PTR 01H
0000:00FF80A4   C6 06 92 02 A0                          MOV IDT_PTR+2,BYTE PTR A0H
0000:00FF80A9   C6 06 93 02 00                          MOV IDT_PTR+3,BYTE PTR 00H
0000:00FF80AE   C6 06 94 02 00                          MOV IDT_PTR+4,BYTE PTR 00H
0000:00FF80B3   C6 06 95 02 00                          MOV IDT_PTR+5,BYTE PTR 00H
0000:00FF80B8   3E 0F 01 16 88 02               LGDT DS:GDT_PTR                         ;PUT GDT_PTR IN GDTR
0000:00FF80BE   3E 0F 01 1E 90 02               LIDT DS:IDT_PTR                         ;PUT IDT_PTR IN IDTR
0000:00FF80C4   0F 01 E0                            SMSW AX                             ;GET STATUS WORD
0000:00FF80C7   0C 01                               OR AL,01H                          ;SET PROTECTED MODE BIT
0000:00FF80C9   0F 01 F0                            LMSW AX     ;LOAD STATUS WORD AND BEGIN PROTECTED MODE
0000:00FF80CC   EB 00                               JMP SHORT FLUSH                     ;FLUSH PREFETCH QUEUE
0000:00FF80CE   B8 10 00            FLUSH:  MOV AX,0010H                               ;SET SEGMENT REGISTERS TO
0000:00FF80D1   8E D0                               MOV SS,AX                          ;DATA DESCRIPTOR SELECTOR
0000:00FF80D3   8E D8                               MOV DS,AX
0000:00FF80D5   8E C0                               MOV ES,AX
0000:00FF80D7   8E E0                               MOV FS,AX
0000:00FF80D9   8E E8                               MOV GS,AX
                                    ;SET UP INSTRUCTION
0000:00FF80DB   66                                  DB 66H                              ;JMP FAR SET_ESP
0000:00FF80DC   EA                                  DB EAH                              ;SET CS TO 08H
                                    ;-----------------------------------------------------------
0000:00FF80DD   E380                                DW 80E3H
0000:00FF80DF   FF00                                DW 00FFH
0000:00FF80E1   0800                                DW 0008H
0000:00FF80E3   BC 00 80 00 00      SET_ESP:        MOV ESP,00008000H
0000:00FF80E8   FC                                  CLD

0000:00FF80E9   A1 CA 02 00 00                      MOV EAX,POWER_UP
                                              DB DATA_SZ
0000:00FF80EE   66
0000:00FF80EF   3D 55 55                            CMP AX,05555H
0000:00FF80F2   0F 84 28 00 00 00                   JZ INIT_SYS_TASK

0000:00FF80F8   BE 53 81 FF 00                      MOV ESI,OFFSET NULL_DESC0           ;INITALIZE THE REST OF THE
0000:00FF80FD   BF 00 00 00 00                      MOV EDI,OFFSET NULL_G_DESC          ;SYSTEM RAM
0000:00FF8102   B9 CC 02 00 00                      MOV ECX,000002CCH
0000:00FF8107   3E F3 A4                            REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF810A   BF 00 03 00 00                      MOV EDI,00000300H
0000:00FF810F   B9 01 7C 00 00                      MOV ECX,00007C01H
0000:00FF8114   33 C0                               XOR EAX,EAX

0000:00FF8116   88 07               CLEAR_RAM:      MOV [EDI],AL
0000:00FF8118   47                                  INC EDI
0000:00FF8119   E2 FB                               LOOP CLEAR_RAM
0000:00FF811B   E9 24 00 00 00                      JMP SW_SYS_TASK

0000:00FF8120   BE 53 81 FF 00      INIT_SYS_TASK:  MOV ESI,OFFSET NULL_DESC0
0000:00FF8125   BF 00 00 00 00                      MOV EDI,00H
0000:00FF812A   B9 30 02 00 00                      MOV ECX,230H
0000:00FF812F   3E F3 A4                            REP MOVSB DS:[EDI],CS:[ESI]

0000:00FF8132   BE 03 83 FF 00                MOV ESI,OFFSET INI_SYS_BK_LNK
0000:00FF8137   BF B0 01 00 00                MOV EDI,OFFSET SYS_BACKLINK
0000:00FF813C   B9 68 00 00 00                MOV ECX,OFFSET DEBUG_EXCEP
0000:00FF8141   3E F3 A4                      REP MOVSB DS:[EDI],CS:[ESI]

0000:00FF8144   90                  SW_SYS_TASK:    NOP
0000:00FF8145   66                                  DB DATA_SZ                          ;LOAD TR WITH USER
0000:00FF8146   B9 28 00                            MOV CX,0028H                        ;TSS DESCRIPTOR
0000:00FF8149   0F 00 D9                            LTR CX

0000:00FF814C   9A                                  DB 9AH                              ;CALL SYSTEM TASK
0000:00FF814D   0000                                DW 0000H
0000:00FF814F   0000                                DW 0000H
0000:00FF8151   2000                                DW 0020H
                                    ;*****************************************************************
                                    ;*                                                               *
```



```
                                        ;* SYSTEM DATA AND CONSTANTS                                    *
                                        ;*                                                             *
                                        ;*****************************************************************
                                        ;SYSTEM CONSTANTS & ADDRESSES
                                        ;GDT ENTRIES
                                        ;-------------------------------------------------------------------------------
0000:00FF8153   00 00 00 00 00 00       NULL_DESC0:     DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF8159   00 00
0000:00FF815B   FF FF 00 00 00 9F       CODE_DESC0:     DB FFH,FFH,00H,00H,00H,9FH,CFH,00H
0000:00FF8161   CF 00
0000:00FF8163   FF FF 00 00 00 93       DATA_DESC0:     DB FFH,FFH,00H,00H,00H,93H,CFH,00H
0000:00FF8169   CF 00
0000:00FF816B   4F 00 50 00 00 82       LDT_DESC:       DB 4FH,00H,50H,00H,00H,82H,40H,00H
0000:00FF8171   40 00
0000:00FF8173   67 00 B0 01 00 89       SYS_TASK_DESC:  DB 67H,00H,B0H,01H,00H,89H,00H,00H
0000:00FF8179   00 00
0000:00FF817B   67 00 20 02 00 89       USER_TASK_DESC: DB 67H,00H,20H,02H,00H,89H,00H,00H
0000:00FF8181   00 00
0000:00FF8183   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF8189   00 00
0000:00FF818B   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF8191   00 00
0000:00FF8193   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF8199   00 00
0000:00FF819B   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF81A1   00 00

                                      ;-------------------------------------------------------------------------

0000:00FF81A3                           ORG 0FF81A3H
                                      ;LDT ENTRIES
                                      ;-------------------------------------------------------------------------
0000:00FF81A3   00 00 00 00 00 00       NULL_DESC1:     DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF81A9   00 00
0000:00FF81AB   FF FF 00 00 00 9F       CODE_DESC1:     DB FFH,FFH,00H,00H,00H,9FH,CFH,00H
0000:00FF81B1   CF 00
0000:00FF81B3   FF FF 00 00 00 93       DATA_DESC1:     DB FFH,FFH,00H,00H,00H,93H,CFH,00H
0000:00FF81B9   CF 00
0000:00FF81BB   00 00 00 00 00 00       USER_DECS1:     DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF81C1   00 00
0000:00FF81C3   00 00 00 00 00 00       USER_DECS2:     DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF81C9   00 00
0000:00FF81CB   00 00 00 00 00 00       USER_DESC3:     DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF81D1   00 00

                                      ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF81D3   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H
0000:00FF81D9   00 00                     DB 00H,00H
0000:00FF81DB   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H
0000:00FF81E1   00 00                     DB 00H,00H
0000:00FF81E3   00 00 00 00 00 00         DB 00H,00H,00H,00H,00H,00H
0000:00FF81E9   00 00                     DB 00H,00H
0000:00FF81EB   FF F7 00 00 00 7E         DB FFH,F7H,00H,00H,00H,7EH
0000:00FF81F1   00 00                     DB 00H,00H
0000:00FF81F3   00 87 08 00 00 8E         DB 00H,87H,08H,00H,00H,8EH
0000:00FF81F9   FF 00                     DB FFH,00H
                                      ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                                        ;IDT ENTRIES:
                                      ;-------------------------------------------------------------------------
0000:00FF81FB   00 00 20 00 00 85       INT_00_DESC:    DB 00H,00H,20H,00H,00H,85H,00H,00H
0000:00FF8201   00 00

                                      ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

0000:00FF8203   E3 80 08 00 00 8E         DB E3H,80H,08H,00H,00H,8EH,FFH,00H
0000:00FF8209   FF 00
0000:00FF820B   25 87 08 00 00 8E         DB 25H,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8211   FF 00
0000:00FF8213   4A 87 08 00 00 8E         DB 4AH,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8219   FF 00
0000:00FF821B   6F 87 08 00 00 8E         DB 6FH,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8221   FF 00
0000:00FF8223   94 87 08 00 00 8E         DB 94H,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8229   FF 00
0000:00FF822B   B9 87 08 00 00 8E         DB B9H,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8231   FF 00
0000:00FF8233   DE 87 08 00 00 8E         DB DEH,87H,08H,00H,00H,8EH,FFH,00H
0000:00FF8239   FF 00
0000:00FF823B   03 88 08 00 00 8E         DB 03H,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8241   FF 00
0000:00FF8243   28 88 08 00 00 8E         DB 28H,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8249   FF 00
0000:00FF824B   4D 88 08 00 00 8E         DB 4DH,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8251   FF 00
0000:00FF8253   72 88 08 00 00 8E         DB 72H,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8259   FF 00
```



```
0000:00FF825B   97 88 08 00 00 8E            DB 97H,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8261   FF 00
0000:00FF8263   BC 88 08 00 00 8E            DB BCH,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8269   FF 00
0000:00FF826B   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF8271   FF FF
0000:00FF8273   E1 88 08 00 00 8E            DB E1H,88H,08H,00H,00H,8EH,FFH,00H
0000:00FF8279   FF 00
0000:00FF827B   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF8281   FF FF
0000:00FF8283   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF8289   FF FF
0000:00FF828B   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF8291   FF FF
0000:00FF8293   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF8299   FF FF
0000:00FF829B   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82A1   FF FF
0000:00FF82A3   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82A9   FF FF
0000:00FF82AB   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82B1   FF FF
0000:00FF82B3   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82B9   FF FF
0000:00FF82BB   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82C1   FF FF
0000:00FF82C3   FF FF FF FF FF FF            DB FFH,FFH,FFH,FFH,FFH,FFH,FFH,FFH
0000:00FF82C9   FF FF

0000:00FF82F3                                ORG 00FF82F3H
0000:00FF82F3   00 00 00 00 00 00            DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF82F9   00 00
0000:00FF82FB   00 00 00 00 00 00            DB 00H,00H,00H,00H,00H,00H,00H,00H
0000:00FF8301   00 00

                                     ;--------------------------------------------------------------------------
                                      ;SYSTEM TSS ENTRIES
                                     ;--------------------------------------------------------------------------
0000:00FF8303   0000                 INI_SYS_BK_LNK: DW 0000H
0000:00FF8305   0000                                 DW 0000H
0000:00FF8307   0080 0000            INI_SYS_ESP0:   DW 8000H,0000H
0000:00FF830B   1000                 INI_SYS_SS0:    DW 0010H
0000:00FF830D   0000                                 DW 0000H
0000:00FF830F   0080 0000            INI_SYS_ESP1:   DW 8000H,0000H
0000:00FF8313   1000                 INI_SYS_SS1:    DW 0010H
0000:00FF8315   0000                                 DW 0000H
0000:00FF8317   0080 0000            INI_SYS_ESP2:   DW 8000H,0000H
0000:00FF831B   1000                 INI_SYS_SS2:    DW 0010H
0000:00FF831D   0000                                 DW 0000H
0000:00FF831F   0000 0000            INI_SYS_CR3:    DW 0000H,0000H
0000:00FF8323   A08A FF00            INI_SYS_EIP:    DW 8AA0H,00FFH
0000:00FF8327   0200 0000            INI_SYS_EFLAGS: DW 0002H,0000H
0000:00FF832B   0000 0000            INI_SYS_EAX:    DW 0000H,0000H
0000:00FF832F   0000 0000            INI_SYS_ECX:    DW 0000H,0000H
0000:00FF8333   0000 0000            INI_SYS_EDX:    DW 0000H,0000H
0000:00FF8337   0000 0000            INI_SYS_EBX:    DW 0000H,0000H
0000:00FF833B   0080 0000            INI_SYS_ESP:    DW 8000H,0000H
0000:00FF833F   0000 0000            INI_SYS_EBP:    DW 0000H,0000H
0000:00FF8343   0000 0000            INI_SYS_ESI:    DW 0000H,0000H
0000:00FF8347   0000 0000            INI_SYS_EDI:    DW 0000H,0000H
0000:00FF834B   1000                 INI_SYS_ES:     DW 0010H
0000:00FF834D   0000                                 DW 0000H
0000:00FF834F   0800                 INI_SYS_CS:     DW 0008H
0000:00FF8351   0000                                 DW 0000H
0000:00FF8353   1000                 INI_SYS_SS:     DW 0010H
0000:00FF8355   0000                                 DW 0000H
0000:00FF8357   1000                 INI_SYS_DS:     DW 0010H
0000:00FF8359   0000                                 DW 0000H
0000:00FF835B   1000                 INI_SYS_FS:     DW 0010H
0000:00FF835D   0000                                 DW 0000H
0000:00FF835F   1000                 INI_SYS_GS:     DW 0010H
0000:00FF8361   0000                                 DW 0000H
0000:00FF8363   0000                 INI_SYS_LDT:    DW 0000H
0000:00FF8365   0000                                 DW 0000H
0000:00FF8367   0000                 INI_SYS_T:      DW 0000H
0000:00FF8369   6800                 INI_SYS_IO:     DW 0068H

                                     ;--------------------------------------------------------------------------
                                     ;MONITOR DATA
                                     ;--------------------------------------------------------------------------
0000:00FF836B   4400 AA00                    INIT_FREQ:      DW 0044H,00AAH
0000:00FF836F   0100 0000                    INIT_DURATION:  DW 0001H,0000H

                                     ;--------------------------------------------------------------------------
                                     ;USER TSS ENTRIES
                                     ;--------------------------------------------------------------------------
0000:00FF8373   0000                 INI_USR_BK_LNK: DW 0000H
0000:00FF8375   0000                                 DW 0000H
0000:00FF8377   007F 0000            INI_USR_ESP0:   DW 7F00H,0000H
0000:00FF837B   1000                 INI_USR_SS0:    DW 0010H
0000:00FF837D   0000                                 DW 0000H
0000:00FF837F   007F 0000            INI_USR_ESP1:   DW 7F00H,0000H
0000:00FF8383   1000                 INI_USR_SS1:    DW 0010H
0000:00FF8385   0000                                 DW 0000H
0000:00FF8387   007F 0000            INI_USR_ESP2:   DW 7F00H,0000H
0000:00FF838B   1000                 INI_USR_SS2:    DW 0010H
0000:00FF838D   0000                                 DW 0000H
```

```
0000:00FF838F   0000 0000               INI_USR_CR3:    DW 0000H,0000H
0000:00FF8393   0003 0000               INI_USR_EIP:    DW 0300H,0000H
0000:00FF8397   0200 0000               INI_USR_EFLAGS: DW 0002H,0000H
0000:00FF839B   0000 0000               INI_USR_EAX:    DW 0000H,0000H
0000:00FF839F   0000 0000               INI_USR_ECX:    DW 0000H,0000H
0000:00FF83A3   0000 0000               INI_USR_EDX:    DW 0000H,0000H
0000:00FF83A7   0000 0000               INI_USR_EBX:    DW 0000H,0000H
0000:00FF83AB   007F 0000               INI_USR_ESP:    DW 7F00H,0000H
0000:00FF83AF   0000 0000               INI_USR_EBP:    DW 0000H,0000H
0000:00FF83B3   0000 0000               INI_USR_ESI:    DW 0000H,0000H
0000:00FF83B7   0000 0000               INI_USR_EDI:    DW 0000H,0000H
0000:00FF83BB   1000                    INI_USR_ES:     DW 0010H
0000:00FF83BD   0000                                    DW 0000H
0000:00FF83BF   0800                    INI_USR_CS:     DW 0008H
0000:00FF83C1   0000                                    DW 0000H
0000:00FF83C3   1000                    INI_USR_SS:     DW 0010H
0000:00FF83C5   0000                                    DW 0000H
0000:00FF83C7   1000                    INI_USR_DS:     DW 0010H
0000:00FF83C9   0000                                    DW 0000H
0000:00FF83CB   1000                    INI_USR_FS:     DW 0010H
0000:00FF83CD   0000                                    DW 0000H
0000:00FF83CF   1000                    INI_USR_GS:     DW 0010H
0000:00FF83D1   0000                                    DW 0000H
0000:00FF83D3   0000                    INI_USR_LDT:    DW 0000H
0000:00FF83D5   0000                                    DW 0000H
0000:00FF83D7   0000                    INI_USR_T:      DW 0000H
0000:00FF83D9   6800                    INI_USR_IO:     DW 0068H

                                ;-------------------------------------------------------------------------
                                ;POINTER DATA
                                ;-------------------------------------------------------------------------
0000:00FF83DB   2F 00 00 00 00 00       GDT_PTR_DATA:           DB 2FH,00H,00H,00H,00H,00H,00H,00H
0000:00FF83E1   00 00
0000:00FF83E3   0F 01 A0 00 00 00       IDT_PTR_DATA:           DB 0FH,01H,A0H,00H,00H,00H,00H,00H
0000:00FF83E9   00 00
0000:00FF83EB   FF 00 00 00             INI_LAST_KEY:           DB FFH,00H,00H,00H
0000:00FF83EF   13 07 80 BB             INI_SYS_FLAGS:          DB 13H,07H,80H,BBH
0000:00FF83F3   00 03 00 00             INI_CURRENT_ADDRESS:    DB 00H,03H,00H,00H
0000:00FF83F7   00 00 00 00             INI_DATA_MODE:          DB 00H,00H,00H,00H
0000:00FF83FB   00 00 00 00             INI_START_ADDRESS:      DB 00H,00H,00H,00H
0000:00FF83FF   00 00 00 00             INI_END_ADDRESS:        DB 00H,00H,00H,00H
0000:00FF8403   00 00 00 00             INI_DEST_ADDRESS:       DB 00H,00H,00H,00H
0000:00FF8407   00 00 00 00             INI_DR0:                DB 00H,00H,00H,00H
0000:00FF840B   00 00 00 00             INI_DR1:                DB 00H,00H,00H,00H
0000:00FF840F   00 00 00 00             INI_DR2:                DB 00H,00H,00H,00H
0000:00FF8413   00 00 00 00             INI_DR3:                DB 00H,00H,00H,00H
0000:00FF8417   00 00 00 00             INI_DR6:                DB 00H,00H,00H,00H
0000:00FF841B   00 00 55 55             INI_POWER_UP:           DB 00H, 00H,55H,55H

                                ;-------------------------------------------------------------------------
                                ;DISPLAY CODES FOR LCD DISPLAY MESSAGES
                                ;-------------------------------------------------------------------------
0000:00FF841F   30 31 32 33 34 35       HEXCODES:       DB 30H,31H,32H,33H,34H,35H,36H,37H
0000:00FF8425   36 37
0000:00FF8427   38 39 41 42 43 44                       DB 38H,39H,41H,42H,43H,44H,45H,46H      ;DISPLAY CODES
0000:00FF842D   45 46

                                ;-------------------------------------------------------------------------
                                                        ;FOR HEX DIGITS
                                ;-------------------------------------------------------------------------
0000:00FF842F   53 68 69 66 74 20       SHIFT_MSG:      DB 53H,68H,69H,66H,74H,20H,20H,20H
0000:00FF8435   20 20
0000:00FF8437   20 20 20 20 20 20   BLANK:              DB 20H,20H,20H,20H,20H,20H,20H,20H,20H
0000:00FF843D   20 20 20

0000:00FF8440   20 20 20 20 20 20                   DB 20H,20H,20H,20H,20H,20H
0000:00FF8446   20 20                               DB 20H,20H

0000:00FF8448   53 4C 38 30 33 38       READY:          DB 53H,4CH,38H,30H,33H,38H,36H,20H
0000:00FF844E   36 20
0000:00FF8450   E4 4C 61 62 20 20                       DB E4H,4CH,61H,62H,20H,20H,20H,20H,20H
0000:00FF8456   20 20 20
0000:00FF8459   78 78 78 78 78 78       X_MSG:          DB 78H,78H,78H,78H,78H,78H,78H,78H
0000:00FF845F   78 78

0000:00FF8461   46 69 6C 65 B0 B0           DB 46H,69H,6CH,65H,B0H,B0H
0000:00FF8467   B0 B0                       DB B0H,B0H

0000:00FF8469   53 74 61 72 74 B0           DB 53H,74H,61H,72H,74H,B0H
0000:00FF846F   B0 B0 45 6E 64 B0           DB B0H,B0H,45H,6EH,64H,B0H,B0H,B0H
0000:00FF8475   B0 B0
0000:00FF8477   B0 B0 46 69 6C 65           DB B0H,B0H,46H,69H,6CH,65H,20H,45H
0000:00FF847D   20 45
0000:00FF847F   72 72 6F 72 20 20           DB 72H,72H,6FH,72H,20H,20H,20H,20H
0000:00FF8485   20 20
```

```
0000:00FF8487   20 20 20                        DB 20H,20H,20H
0000:00FF848A   4F 75 74 73 69      MESSAGE_1:          DB 4FH,75H,74H,73H,69H
0000:00FF848F   64 65 20 55 73 65               DB 64H,65H,20H,55H,73H,65H,72H,20H
0000:00FF8495   72 20
0000:00FF8497   52 41 4D 20                     DB 52H,41H,4DH,20H
0000:00FF849B   41 64 64 72         MESSAGE_2:          DB 41H,64H,64H,72H
0000:00FF849F   65 73 72 20 4F 72               DB 65H,73H,72H,20H,4FH,72H,64H,65H
0000:00FF84A5   64 65
0000:00FF84A7   72 20 20 20 20 4F               DB 72H,20H,20H,20H,20H,4FH,66H,66H
0000:00FF84AD   66 66
0000:00FF84AF   73 65 74 20 54 6F               DB 73H,65H,74H,20H,54H,6FH,6FH,20H
0000:00FF84B5   6F 20


                                        START_MSG:      ;DB 53H,74H,61H,72H,74H,B0H,B0H,B0H

                                        END_MSG:        ;DB 45H,6EH,64H,B0H,B0H,B0H,B0H,B0H

                                ;-------------------------------------------------------------------------
0000:00FF84B7   4C 61                           DB 4CH,61H
0000:00FF84B9   72 67 65 20                     DB 72H,67H,65H,20H
                                ;-------------------------------------------------------------------------


0000:00FF84BD                           BOUNDS_ERROR_MSG:

0000:00FF84BD
0000:00FF84BD   44 65 73 74 B0 B0       DEST_MSG:       DB 44H,65H,73H,74H,B0H,B0H,B0H,B0H
0000:00FF84C3   B0 B0

0000:00FF84C5
0000:00FF84C5   42 72 6B 50 74 20       BRPT_MSG:       DB 42H,72H,6BH,50H,74H,20H,B0H,20H
0000:00FF84CB   B0 20

0000:00FF84CD
0000:00FF84CD   43 52 33 B0 B0 B0       REG_32_MSG:     DB 43H,52H,33H,B0H,B0H,B0H,B0H,B0H
0000:00FF84D3   B0 B0

0000:00FF84D5   45 49 50 B0 B0 B0                       DB 45H,49H,50H,B0H,B0H,B0H,B0H,B0H
0000:00FF84DB   B0 B0

0000:00FF84DD   45 46 4C 41 47 53                       DB 45H,46H,4CH,41H,47H,53H,B0H,B0H
0000:00FF84E3   B0 B0

0000:00FF84E5   45 41 58 B0 B0 B0                       DB 45H,41H,58H,B0H,B0H,B0H,B0H,B0H
0000:00FF84EB   B0 B0

0000:00FF84ED   45 43 58 B0 B0 B0                       DB 45H,43H,58H,B0H,B0H,B0H,B0H,B0H
0000:00FF84F3   B0 B0

0000:00FF84F5   45 44 58 B0 B0 B0                       DB 45H,44H,58H,B0H,B0H,B0H,B0H,B0H
0000:00FF84FB   B0 B0

0000:00FF84FD   45 42 58 B0 B0 B0                       DB 45H,42H,58H,B0H,B0H,B0H,B0H,B0H
0000:00FF8503   B0 B0

0000:00FF8505   45 53 50 B0 B0 B0                       DB 45H,53H,50H,B0H,B0H,B0H,B0H,B0H
0000:00FF850B   B0 B0

0000:00FF850D   45 42 50 B0 B0 B0                       DB 45H,42H,50H,B0H,B0H,B0H,B0H,B0H
0000:00FF8513   B0 B0

0000:00FF8515   45 53 49 B0 B0 B0                       DB 45H,53H,49H,B0H,B0H,B0H,B0H,B0H
0000:00FF851B   B0 B0

0000:00FF851D   45 44 49 B0 B0 B0                       DB 45H,44H,49H,B0H,B0H,B0H,B0H,B0H
0000:00FF8523   B0 B0

0000:00FF8525
0000:00FF8525   45 53 B0 B0 B0 B0       REG_16_MSG:     DB 45H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF852B   B0 B0

0000:00FF852D   43 53 B0 B0 B0 B0                       DB 43H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF8533   B0 B0

0000:00FF8535   53 53 B0 B0 B0 B0                       DB 53H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF853B   B0 B0

0000:00FF853D   44 53 B0 B0 B0 B0                       DB 44H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF8543   B0 B0

0000:00FF8545   46 53 B0 B0 B0 B0                       DB 46H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF854B   B0 B0

0000:00FF854D   47 53 B0 B0 B0 B0                       DB 47H,53H,B0H,B0H,B0H,B0H,B0H,B0H
0000:00FF8553   B0 B0

0000:00FF8555   4C 44 54 B0 B0 B0                       DB 4CH,44H,54H,B0H,B0H,B0H,B0H,B0H
0000:00FF855B   B0 B0


0000:00FF855D   5A 65 72 6F 20 44   I_D_1:          DB 5AH,65H,72H,6FH,20H,44H,69H,76H
0000:00FF8563   69 76
0000:00FF8565   64 65 20 45 72 72           DB 64H,65H,20H,45H,72H,72H,6FH,72H,20H
0000:00FF856B   6F 72 20
0000:00FF856E   42 72 65 61 6B 70   I_D_2:    DB 42H,72H,65H,61H,6BH,70H,6FH
0000:00FF8574   6F
0000:00FF8575   69 6E 74 20 49 6E           DB 69H,6EH,74H,20H,49H,6EH,73H,74H
```



```
     0000:00FF857B   73 74
690  0000:00FF857D   72 20                        DB 72H,20H
691  0000:00FF857F   4F 76 65 72 66 6C  I_D_3:    DB 4FH,76H,65H,72H,66H,6CH
692  0000:00FF8585   6F 77 2C 20 49 4E            DB 6FH,77H,2CH,20H,49H,4EH,54H,4FH,20H
     0000:00FF858B   54 4F 20
693  0000:00FF858E   49 6E                        DB 49H,6EH
694  0000:00FF8590   42 6F 75 6E 64     I_D_4:    DB 42H,6FH,75H,6EH,64H
695  0000:00FF8595   73 20 43 68 65 63            DB 73H,20H,43H,68H,65H,63H,6BH,20H
     0000:00FF859B   6B 20
696  0000:00FF859D   20 20 20 20                  DB 20H,20H,20H,20H
697  0000:00FF85A1   49 6E 76 61        I_D_5:    DB 49H,6EH,76H,61H
698  0000:00FF85A5   6C 69 64 20 4F 70            DB 6CH,69H,64H,20H,4FH,70H,63H,6FH
     0000:00FF85AB   63 6F
699  0000:00FF85AD   64 65 20 20 20               DB 64H,65H,20H,20H,20H
700  0000:00FF85B2   38 30 33           I_D_6:    DB 38H,30H,33H
701  0000:00FF85B5   38 37 20 4E 6F 74            DB 38H,37H,20H,4EH,6FH,74H,20H,41H
     0000:00FF85BB   20 41
702  0000:00FF85BD   76 61 69 6C 61 62            DB 76H,61H,69H,6CH,61H,62H
703  0000:00FF85C3   44 6F              I_D_7:    DB 44H,6FH
704  0000:00FF85C5   75 62 6C 65 20 46            DB 75H,62H,6CH,65H,20H,46H,61H,75H
     0000:00FF85CB   61 75
705  0000:00FF85CD   6C 74 20 20 20 20            DB 6CH,74H,20H,20H,20H,20H,20H
     0000:00FF85D3   20
706  0000:00FF85D4   38                 I_D_8:    DB 38H
707  0000:00FF85D5   30 33 38 37 20 53            DB 30H,33H,38H,37H,20H,53H,65H,67H
     0000:00FF85DB   65 67
708  0000:00FF85DD   20 4F                        DB 20H,4FH
709  0000:00FF85DF   76 65 72 72 75 6E            DB 76H,65H,72H,72H,75H,6EH
710  0000:00FF85E5   49 6E              I_D_9:    DB 49H,6EH
711  0000:00FF85E7   76 61 6C 69 64 20            DB 76H,61H,6CH,69H,64H,20H,54H,53H
     0000:00FF85ED   54 53
712  0000:00FF85EF   53 20 20 20 20 20            DB 53H,20H,20H,20H,20H,20H,20H
     0000:00FF85F5   20
713  0000:00FF85F6   53                 I_D_10:   DB 53H
714  0000:00FF85F7   65 67 20 4C 6F 74            DB 65H,67H,20H,4CH,6FH,74H,20H,50H
     0000:00FF85FD   20 50
715  0000:00FF85FF   72 65 73 65 6C 74            DB 72H,65H,73H,65H,6CH,74H,20H,20H
     0000:00FF8605   20 20
716  0000:00FF8607   53 74 61 63 6B 20  I_D_11:   DB 53H,74H,61H,63H,6BH,20H,45H,78H
     0000:00FF860D   45 78
717  0000:00FF860F   63 65 70 74 69 6F            DB 63H,65H,70H,74H,69H,6FH,6EH,20H,20H
     0000:00FF8615   6E 20 20
718  0000:00FF8618   67 50 20 45 78 63  I_D_12:   DB 67H,50H,20H,45H,78H,63H,65H
     0000:00FF861E   65
719  0000:00FF861F   70 74 69 6F 6E 20            DB 70H,74H,69H,6FH,6EH,20H,20H,20H
     0000:00FF8625   20 20
720  0000:00FF8627   20 20                        DB 20H,20H
721  0000:00FF8629   50 61 67 65 20 46  I_D_13:   DB 50H,61H,67H,65H,20H,46H
722  0000:00FF862F   61 75 6C 74 20 20            DB 61H,75H,6CH,74H,20H,20H,20H,20H
     0000:00FF8635   20 20
723  0000:00FF8637   20 20 20                     DB 20H,20H,20H
724  0000:00FF863A   38 30 33 38 37     I_D_14:   DB 38H,30H,33H,38H,37H
725  0000:00FF863F   20 45 72 72 6F 72            DB 20H,45H,72H,72H,6FH,72H,20H,20H
     0000:00FF8645   20 20
726  0000:00FF8647   20 20 20 20                  DB 20H,20H,20H,20H
727
728                                  ;---------------------------------------------------------------------
729                                  ;
730                                  ;NOTE: The SDK-386 monitor uses a flat model.  The addresses required
731                                  ;are the full 32 bit addresses that are sometimes difficult to obtain
732                                  ;from certain types of assemblers.  All addresses in the CALL TABLE are
733                                  ;generated by a technique that may cause error statements from the
734                                  ;assembler although the assembled result is in fact correct.  This
735                                  ;is the case in the following part of the listing.  The error messages
736                                  ;have not been edited out of the listing because we wish to be able
737                                  ;to regenerate any listing without modification of the assembler
738                                  ;output.  This is the case in the following CALL TABLE.
739                                  ;
740                                  ;---------------------------------------------------------------------
741
742  0000:00FF864B   EC8B FF00           CALLTABLE:    DW  NULL_ROUTINE,TOP_ROM          ;SHIFT (NOT RETURNED)
     ***** # TOO LARGE *****
743  0000:00FF864F   818B FF00                         DW  RUN_USER_A,TOP_ROM            ;RUN
     ***** # TOO LARGE *****
744  0000:00FF8653   AC95 FF00                         DW  BREAK_POINT,TOP_ROM           ;CBP
     ***** # TOO LARGE *****
745  0000:00FF8657   058D FF00                         DW  DECREMENT,TOP_ROM             ;-
     ***** # TOO LARGE *****
746  0000:00FF865B
747  0000:00FF865B   208E FF00                         DW UP_LOAD,TOP_ROM                ;UPLD
     ***** # TOO LARGE *****
748  0000:00FF865F   EC8B FF00                         DW NULL_ROUTINE,TOP_ROM           ;STEP
     ***** # TOO LARGE *****
749  0000:00FF8663   C88C FF00                         DW LWORD_MODE,TOP_ROM             ;DWD
     ***** # TOO LARGE *****
750  0000:00FF8667   E08C FF00                         DW INCREMENT,TOP_ROM              ;+
     ***** # TOO LARGE *****
751  0000:00FF866B
752  0000:00FF866B   B88E FF00                         DW DN_LOAD,TOP_ROM                ;DNLD
     ***** # TOO LARGE *****
753  0000:00FF866F   F692 FF00                         DW INSERT,TOP_ROM                 ;INS
     ***** # TOO LARGE *****
754  0000:00FF8673   BE8C FF00                         DW WORD_MODE,TOP_ROM              ;WORD
     ***** # TOO LARGE *****
755  0000:00FF8677   4694 FF00                         DW TASK_S_32,TOP_ROM              ;TS32
     ***** # TOO LARGE *****
756  0000:00FF867B
757  0000:00FF867B   5B8D FF00                         DW MOVE_DATA,TOP_ROM              ;MOV
```



```
      ***** # TOO LARGE *****
758   0000:00FF867F   2A8D FF00                    DW AUTO,TOP_ROM                ;AUTO
      ***** # TOO LARGE *****
759   0000:00FF8683   B48C FF00                    DW BYTE_MODE,TOP_ROM           ;BYTE
      ***** # TOO LARGE *****
760   0000:00FF8687   7C8C FF00                    DW ADDRESS,TOP_ROM             ;ADDR
      ***** # TOO LARGE *****
761   0000:00FF868B
762   0000:00FF868B   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM    ;SHIFT, SHIFT (NOT RETURNED)
      ***** # TOO LARGE *****
763   0000:00FF868F   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, RUN
      ***** # TOO LARGE *****
764   0000:00FF8693   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SBP
      ***** # TOO LARGE *****
765   0000:00FF8697   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, -
      ***** # TOO LARGE *****
766   0000:00FF869B
767   0000:00FF869B   9B90 FF00                    DW TAPE_WRT,TOP_ROM            ;TPWT
      ***** # TOO LARGE *****
768   0000:00FF869F   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, STEP
      ***** # TOO LARGE *****
769   0000:00FF86A3   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, DWD
      ***** # TOO LARGE *****
770   0000:00FF86A7   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, +
      ***** # TOO LARGE *****
771   0000:00FF86AB
772   0000:00FF86AB   AE91 FF00                    DW TAPE_RD,TOP_ROM             ;TPRD
      ***** # TOO LARGE *****
773   0000:00FF86AF   4593 FF00                    DW DELETE,TOP_ROM              ;DELETE
      ***** # TOO LARGE *****
774   0000:00FF86B3   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, WORD
      ***** # TOO LARGE *****
775   0000:00FF86B7   EE94 FF00                    DW TASK_S_16,TOP_ROM           ;TS16
      ***** # TOO LARGE *****
776   0000:00FF86BB
777
778   0000:00FF86BB   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, MOV
      ***** # TOO LARGE *****
779
780   0000:00FF86BF   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, AUTO
      ***** # TOO LARGE *****
781
782   0000:00FF86C3   EC8B FF00                    DW NULL_ROUTINE,TOP_ROM        ;SHIFT, BYTE
      ***** # TOO LARGE *****
783
784   0000:00FF86C7   8F93 FF00                    DW RELTV,TOP_ROM               ;REL
      ***** # TOO LARGE *****
785
786                                  ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
787
788   0000:00FF8700                          ORG 00FF8700H
789
790   0000:00FF8700   50                     PUSH EAX
791   0000:00FF8701   53                     PUSH EBX
792   0000:00FF8702   66                     DB DATA_SZ
793   0000:00FF8703   8C DB                  MOV BX,DS
794   0000:00FF8705   B8 10 00 00 00         MOV EAX, 010H
795   0000:00FF870A   66                     DB DATA_SZ
796   0000:00FF870B   B8 00 00 00 00         MOV EAX,00H
797   0000:00FF8710   3E A3 CC 02 00 00      MOV DS:B_DATA,EAX
798   0000:00FF8716   B8 5D 85 FF 00         MOV EAX,OFFSET I_D_1
799   0000:00FF871B   A3 B0 02 00 00         MOV DEST_ADDRESS,EAX
800   0000:00FF8720   E9 E1 01 00 00         JMP 01E1H
801
802   0000:00FF8725   50                     PUSH EAX
803   0000:00FF8726   53                     PUSH EBX
804   0000:00FF8727   66                     DB DATA_SZ
805   0000:00FF8728   8C DB                  MOV BX,DS
806   0000:00FF872A   B8 10 00 00 00         MOV EAX, 010H
807   0000:00FF872F   66                     DB DATA_SZ
808   0000:00FF8730   B8 03 00 00 00         MOV EAX,03H
809   0000:00FF8735   3E A3 CC 02 00 00      MOV DS:B_DATA,EAX
810   0000:00FF873B   B8 6E 85 FF 00         MOV EAX,OFFSET I_D_2
811   0000:00FF8740   A3 B0 02 00 00         MOV DEST_ADDRESS,EAX
812   0000:00FF8745   E9 BC 01 00 00         JMP 01BCH
813
814   0000:00FF874A   50                     PUSH EAX
815   0000:00FF874B   53                     PUSH EBX
816   0000:00FF874C   66                     DB DATA_SZ
817   0000:00FF874D   8C DB                  MOV BX,DS
818   0000:00FF874F   B8 10 00 00 00         MOV EAX, 010H
819   0000:00FF8754   66                     DB DATA_SZ
820   0000:00FF8755   B8 04 00 00 00         MOV EAX,04H
821   0000:00FF875A   3E A3 CC 02 00 00      MOV DS:B_DATA,EAX
822   0000:00FF8760   B8 7F 85 FF 00         MOV EAX,OFFSET I_D_3
823   0000:00FF8765   A3 B0 02 00 00         MOV DEST_ADDRESS,EAX
824   0000:00FF876A   E9 97 01 00 00         JMP 0197H
825
826   0000:00FF876F   50                     PUSH EAX
827   0000:00FF8770   53                     PUSH EBX
828   0000:00FF8771   66                     DB DATA_SZ
829   0000:00FF8772   8C DB                  MOV BX,DS
830   0000:00FF8774   B8 10 00 00 00         MOV EAX, 010H
831   0000:00FF8779   66                     DB DATA_SZ
832   0000:00FF877A   B8 05 00 00 00         MOV EAX,05H
833   0000:00FF877F   3E A3 CC 02 00 00      MOV DS:B_DATA,EAX
834   0000:00FF8785   B8 90 85 FF 00         MOV EAX,OFFSET I_D_4
835   0000:00FF878A   A3 B0 02 00 00         MOV DEST_ADDRESS,EAX
```



```
0000:00FF878F   E9 72 01 00 00          JMP 0172H

0000:00FF8794   50                      PUSH EAX
0000:00FF8795   53                      PUSH EBX
0000:00FF8796   66                      DB DATA_SZ
0000:00FF8797   8C DB                   MOV BX,DS
0000:00FF8799   B8 10 00 00 00          MOV EAX, 010H
0000:00FF879E   66                      DB DATA_SZ
0000:00FF879F   B8 06 00 00 00          MOV EAX,06H
0000:00FF87A4   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF87AA   B8 A1 85 FF 00          MOV EAX,OFFSET I_D_5
0000:00FF87AF   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF87B4   E9 4D 01 00 00          JMP 014DH

0000:00FF87B9   50                      PUSH EAX
0000:00FF87BA   53                      PUSH EBX
0000:00FF87BB   66                      DB DATA_SZ
0000:00FF87BC   8C DB                   MOV BX,DS
0000:00FF87BE   B8 10 00 00 00          MOV EAX, 010H
0000:00FF87C3   66                      DB DATA_SZ
0000:00FF87C4   B8 07 00 00 00          MOV EAX,07H
0000:00FF87C9   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF87CF   B8 B2 85 FF 00          MOV EAX,OFFSET I_D_6
0000:00FF87D4   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF87D9   E9 28 01 00 00          JMP 0128H

0000:00FF87DE   50                      PUSH EAX
0000:00FF87DF   53                      PUSH EBX
0000:00FF87E0   66                      DB DATA_SZ
0000:00FF87E1   8C DB                   MOV BX,DS
0000:00FF87E3   B8 10 00 00 00          MOV EAX, 010H
0000:00FF87E8   66                      DB DATA_SZ
0000:00FF87E9   B8 08 00 00 00          MOV EAX,08H
0000:00FF87EE   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF87F4   B8 C3 85 FF 00          MOV EAX,OFFSET I_D_7
0000:00FF87F9   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF87FE   E9 03 01 00 00          JMP 0103H

0000:00FF8803   50                      PUSH EAX
0000:00FF8804   53                      PUSH EBX
0000:00FF8805   66                      DB DATA_SZ
0000:00FF8806   8C DB                   MOV BX,DS
0000:00FF8808   B8 10 00 00 00          MOV EAX, 010H
0000:00FF880D   66                      DB DATA_SZ
0000:00FF880E   B8 09 00 00 00          MOV EAX,09H
0000:00FF8813   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF8819   B8 D4 85 FF 00          MOV EAX,OFFSET I_D_8
0000:00FF881E   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF8823   E9 DE 00 00 00          JMP 00DEH

0000:00FF8828   50                      PUSH EAX
0000:00FF8829   53                      PUSH EBX
0000:00FF882A   66                      DB DATA_SZ
0000:00FF882B   8C DB                   MOV BX,DS
0000:00FF882D   B8 10 00 00 00          MOV EAX, 010H
0000:00FF8832   66                      DB DATA_SZ
0000:00FF8833   B8 0A 00 00 00          MOV EAX,0AH
0000:00FF8838   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF883E   B8 E5 85 FF 00          MOV EAX,OFFSET I_D_9
0000:00FF8843   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF8848   E9 B9 00 00 00          JMP 00B9H

0000:00FF884D   50                      PUSH EAX
0000:00FF884E   53                      PUSH EBX
0000:00FF884F   66                      DB DATA_SZ
0000:00FF8850   8C DB                   MOV BX,DS
0000:00FF8852   B8 10 00 00 00          MOV EAX, 010H
0000:00FF8857   66                      DB DATA_SZ
0000:00FF8858   B8 0B 00 00 00          MOV EAX,0BH
0000:00FF885D   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF8863   B8 F6 85 FF 00          MOV EAX,OFFSET I_D_10
0000:00FF8868   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF886D   E9 94 00 00 00          JMP EX_2

0000:00FF8872   50                      PUSH EAX
0000:00FF8873   53                      PUSH EBX
0000:00FF8874   66                      DB DATA_SZ
0000:00FF8875   8C DB                   MOV BX,DS
0000:00FF8877   B8 10 00 00 00          MOV EAX, 010H
0000:00FF887C   66                      DB DATA_SZ
0000:00FF887D   B8 0C 00 00 00          MOV EAX,0CH
0000:00FF8882   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF8888   B8 07 86 FF 00          MOV EAX,OFFSET I_D_11
0000:00FF888D   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF8892   E9 6F 00 00 00          JMP EX_2

0000:00FF8897   50                      PUSH EAX
0000:00FF8898   53                      PUSH EBX
0000:00FF8899   66                      DB DATA_SZ
0000:00FF889A   8C DB                   MOV BX,DS
0000:00FF889C   B8 10 00 00 00          MOV EAX, 010H
0000:00FF88A1   66                      DB DATA_SZ
0000:00FF88A2   B8 0D 00 00 00          MOV EAX,0DH
0000:00FF88A7   3E A3 CC 02 00 00       MOV DS:B_DATA,EAX
0000:00FF88AD   B8 18 86 FF 00          MOV EAX,OFFSET I_D_12
0000:00FF88B2   A3 B0 02 00 00          MOV DEST_ADDRESS,EAX
0000:00FF88B7   E9 4A 00 00 00          JMP EX_2
```

```
0000:00FF88BC   50                          PUSH EAX
0000:00FF88BD   53                          PUSH EBX
0000:00FF88BE   66                          DB DATA_SZ
0000:00FF88BF   8C DB                       MOV BX,DS
0000:00FF88C1   B8 10 00 00 00              MOV EAX, 010H
0000:00FF88C6   66                          DB DATA_SZ
0000:00FF88C7   B8 0E 00 00 00              MOV EAX,0EH
0000:00FF88CC   3E A3 CC 02 00 00           MOV DS:B_DATA,EAX
0000:00FF88D2   B8 29 86 FF 00              MOV EAX,OFFSET I_D_13
0000:00FF88D7   A3 B0 02 00 00              MOV DEST_ADDRESS,EAX
0000:00FF88DC   E9 25 00 00 00              JMP EX_2
0000:00FF88E1   50                          PUSH EAX
0000:00FF88E2   53                          PUSH EBX
0000:00FF88E3   66                          DB DATA_SZ
0000:00FF88E4   8C DB                       MOV BX,DS
0000:00FF88E6   B8 10 00 00 00              MOV EAX, 010H
0000:00FF88EB   66                          DB DATA_SZ
0000:00FF88EC   B8 10 00 00 00              MOV EAX,10H
0000:00FF88F1   3E A3 CC 02 00 00           MOV DS:B_DATA,EAX
0000:00FF88F7   B8 3A 86 FF 00              MOV EAX,OFFSET I_D_14
0000:00FF88FC   A3 B0 02 00 00              MOV DEST_ADDRESS,EAX
0000:00FF8901   E9 00 00 00 00              JMP EX_2
0000:00FF8906   58                  EX_2:   POP EAX
0000:00FF8907   3E A3 E4 02 00 00           MOV DS:E_DATA,EAX
0000:00FF890D   58                          POP EAX
0000:00FF890E   3E A3 E0 02 00 00           MOV DS:F_DATA,EAX
0000:00FF8914   3E A1 CC 02 00 00           MOV EAX,DS:B_DATA
0000:00FF891A   3D 08 00 00 00              CMP EAX,08H
0000:00FF891F   0F 84 22 00 00 00           JZ L_6
0000:00FF8925   3D 0A 00 00 00              CMP EAX,0AH
0000:00FF892A   0F 8D 10 00 00 00           JGE L_3
0000:00FF8930   B8 FF FF FF FF      L_1:    MOV EAX,FFFFFFFFH
0000:00FF8935   3E A3 DC 02 00 00           MOV DS:G_DATA,EAX
0000:00FF893B   E9 0E 00 00 00              JMP 0EH
0000:00FF8940   3D 0E 00 00 00      L_3:    CMP EAX,0EH
0000:00FF8945   7F E9                       JG L_1
0000:00FF8947   58                  L_6:    POP EAX
0000:00FF8948   3E A3 DC 02 00 00           MOV DS:G_DATA,EAX
0000:00FF894E   58                          POP EAX
0000:00FF894F   3E A3 D8 02 00 00   L_2:    MOV DS:MEM_VAL_2,EAX
0000:00FF8955   58                          POP EAX
0000:00FF8956   3E A3 D4 02 00 00           MOV DS:H_DATA,EAX
0000:00FF895C   58                          POP EAX
0000:00FF895D   3E A3 D0 02 00 00           MOV DS:I_DATA,EAX
0000:00FF8963   50                          PUSH EAX
0000:00FF8964   B8 08 00 00 00              MOV EAX,08H
0000:00FF8969   50                          PUSH EAX
0000:00FF896A   B8 90 89 FF 00              MOV EAX,00FF8990H
0000:00FF896F   50                          PUSH EAX
0000:00FF8970   3E A1 DC 02 00 00           MOV EAX,DS:G_DATA
0000:00FF8976   3D FF FF FF FF              CMP EAX,FFFFFFFFH
0000:00FF897B   0F 84 01 00 00 00           JZ I_BACK
0000:00FF8981   50                          PUSH EAX
0000:00FF8982   CF                  I_BACK: IRET

                                    ;-------------------------------------------------------------------------

0000:00FF8990                               ORG 00FF8990H
0000:00FF8990   B8 00 8A FF 00              MOV EAX,00FF8A00H
0000:00FF8995   3E A3 D0 01 00 00           MOV DS:SYS_EIP,EAX
0000:00FF899B   66                          DB DATA_SZ
0000:00FF899C   8E DB                       MOV DS,BX
                                    ;-------------------------------------------------------------------------
                                    ;SET UP INSTRUCTION
                                    ;-------------------------------------------------------------------------
0000:00FF899E   9A                          DB 9AH
0000:00FF899F   00 00                       DB 00H,00H
0000:00FF89A1   00 00                       DB 00H,00H
0000:00FF89A3   20 00                       DB 20H,00H
                                    ;-------------------------------------------------------------------------

0000:00FF8A00                               ORG 00FF8A00H
0000:00FF8A00   8B 35 B0 01 00 00           MOV ESI,SYS_BACKLINK
0000:00FF8A06   46                          INC ESI
0000:00FF8A07   46                          INC ESI
0000:00FF8A08   8B 06                       DB 8BH,06H
0000:00FF8A0A   46                          INC ESI
0000:00FF8A0B   46                          INC ESI
0000:00FF8A0C   8B 1E                       MOV EBX,[ESI]
0000:00FF8A0E   25 FF FF FF 00              AND EAX,00FFFFFFH
0000:00FF8A13   81 E3 00 00 00 FF           AND EBX,FF000000H
0000:00FF8A19   0B C3                       OR EAX,EBX
0000:00FF8A1B   8B F8                       DB 8BH,F8H
0000:00FF8A1D   83 C7 20                    ADD EDI,20H
0000:00FF8A20   A1 D8 02 00 00              MOV EAX,MEM_VAL_2
0000:00FF8A25   89 07                       MOV [EDI],EAX
0000:00FF8A27   A3 A0 02 00 00              MOV CURRENT_ADDRESS,EAX
0000:00FF8A2C   83 C7 08                    ADD EDI,08H
0000:00FF8A2F   A1 E0 02 00 00              MOV EAX,F_DATA
0000:00FF8A34   89 07                       MOV [EDI],EAX
0000:00FF8A36   83 C7 0C                    ADD EDI,0CH
0000:00FF8A39   A1 E4 02 00 00              MOV EAX,E_DATA
0000:00FF8A3E   89 07                       MOV [EDI],EAX
0000:00FF8A40   83 C7 18                    ADD EDI,18H
0000:00FF8A43   A1 D4 02 00 00              MOV EAX,H_DATA
0000:00FF8A48   25 FF FF 00 00              AND EAX,0FFFFH
0000:00FF8A4D   89 07                       MOV [EDI],EAX
```

```
0000:00FF8A4F   B9 11 00 00 00              MOV ECX,11H
0000:00FF8A54   A1 B0 02 00 00              MOV EAX,DEST_ADDRESS
0000:00FF8A59   8B F0                       MOV ESI,EAX
0000:00FF8A5B   BF EC 02 00 00              MOV EDI,OFFSET DISPBF
0000:00FF8A60   3E F3 A4                    REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF8A63   E9 A1 00 00 00              JMP PRE_COM



0000:00FF8AA0                                   ORG 00FF8AA0H
                                ;*******************************************************************************
                                ;*                                                                             *
                                ;*                    BEGINING OF CODE FOR SYSTEM TASK                         *
                                ;*                                                                             *
                                ;*******************************************************************************
0000:00FF8AA0
0000:00FF8AA0   BE 73 83 FF 00                  MOV ESI,OFFSET INI_USR_BK_LNK     ;INITIALIZE USER TSS
0000:00FF8AA5   BF 20 02 00 00                  MOV EDI,OFFSET USER_BACKLINK
0000:00FF8AAA   B9 68 00 00 00                  MOV ECX,68H
0000:00FF8AAF   3E F3 A4                        REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF8AB2
0000:00FF8AB2   A1 B4 02 00 00                  MOV EAX,DEBUG_REG_0             ;INITIALIZE DEBUG REGIST
0000:00FF8AB7   0F 23 C0                        MOV DR0,EAX
0000:00FF8ABA   A1 B8 02 00 00                  MOV EAX,DEBUG_REG_1
0000:00FF8ABF   0F 23 C8                        MOV DR1,EAX
0000:00FF8AC2   A1 BC 02 00 00                  MOV EAX,DEBUG_REG_2
0000:00FF8AC7   0F 23 D0                        MOV DR2,EAX
0000:00FF8ACA   A1 C0 02 00 00                  MOV EAX,DEBUG_REG_3
0000:00FF8ACF   0F 23 D8                        MOV DR3,EAX
0000:00FF8AD2   A1 C4 02 00 00                  MOV EAX,DEBUG_REG_7
0000:00FF8AD7   0F 23 F8                        MOV DR7,EAX
0000:00FF8ADA
0000:00FF8ADA   B0 90                           MOV AL,90H                  ;INITIALIZE LOW PORTS
0000:00FF8ADC   BA 06 E0 00 00                  MOV EDX,OFFSET CONTR_L      ;PORT A AS INPUT, ALL OTHERS
0000:00FF8AE1   EE                              OUT DX,AL                   ;AS OUTPUT
0000:00FF8AE2
0000:00FF8AE2   B0 8A                           MOV AL,8AH                      ;INITIALIZE HIGH PORTS
0000:00FF8AE4   BA 07 E0 00 00                  MOV EDX,OFFSET CONTR_H          ;PORT B AND UPPER PORT C
0000:00FF8AE9   EE                              OUT DX,AL                       ;INPUT, REST AS OUTPUT
0000:00FF8AEA
0000:00FF8AEA   B0 FF                           MOV AL,FFH
0000:00FF8AEC   BA 05 E0 00 00                  MOV EDX,OFFSET PORTC_H
0000:00FF8AF1   EE                              OUT DX,AL                        ;INITIALIZE TAPE PORTS
0000:00FF8AF2
0000:00FF8AF2   BC 00 80 00 00                  MOV ESP,00008000H
0000:00FF8AF7
0000:00FF8AF7   B9 11 00 00 00          SIGN_ON:    MOV ECX,00000011H               ;LOAD "SDK-386" MESSAGE
0000:00FF8AFC   BE 48 84 FF 00                  MOV ESI,OFFSET READY
0000:00FF8B01   BF EC 02 00 00                  MOV EDI,OFFSET DISPBF
0000:00FF8B06   3E F3 A4                        REP MOVSB DS:[EDI],CS:[ESI]


0000:00FF8B09   BE C8 02 00 00      PRE_COM:    MOV ESI,OFFSET MEM_VAL_3
0000:00FF8B0E   C6 06 FF                        MOV [ESI],BYTE PTR FFH


                                ;-------------------------------------------------------------------------
                                ;-------------------------------------------------------------------------
                                                    ;MAIN PROGRAM
                                ;-------------------------------------------------------------------------
                                ;-------------------------------------------------------------------------
0000:00FF8B11
0000:00FF8B11   E8 5E 0D 00 00      COM:            CALL KEYBOARD                       ;GET KEY

0000:00FF8B16   C6 06 00                    MOV [ESI],BYTE PTR 00H

0000:00FF8B19   3C 0F               KEY_INTR:       CMP AL,0FH                  ;HEX KEY PRESSED?
0000:00FF8B1B   0F 8E 3C 00 00 00                   JLE CHANGE_DATA             ;IF SO GO TO CHANGE_DATA
0000:00FF8B21   3C 1F                               CMP AL,1FH                  ;WAS ADDR  KEY PRESSED?
0000:00FF8B23   0F 84 3B 00 00 00                   JZ CHANGE_ADDRESS           ;IF SO GO TO CHANGE_ADDRESS
0000:00FF8B29   3C 1D                               CMP AL,1DH                  ;WAS  AUTO KEY PRESSED?
0000:00FF8B2B   0F 84 3A 00 00 00                   JZ AUTO_CHANGE              ;IF SO  GO TO AUTO CHANGE
0000:00FF8B31   3C 11                               CMP AL,11H                  ;WAS RUN KEY PRESSED?
0000:00FF8B33   0F 84 48 00 00 00                   JZ RUN_USER_A               ;IF SO GO TO RUN USER
0000:00FF8B39   3C 15                               CMP AL,15H                  ;WAS STEP KEY PRESSED?
0000:00FF8B3B   0F 84 31 00 00 00                   JZ  SINGLE_STEP             ;IF SO GO TO SINGLE_STEP
0000:00FF8B41   BE 4B 86 FF 00                      MOV ESI,OFFSET CALLTABLE    ;GET CALLTABLE ADDRESS
0000:00FF8B46   3C 2F                               CMP AL,2FH                  ;WAS KEY INVALID?
0000:00FF8B48   7F BF                               JG PRE_COM                  ;IF SO GO TO COM
0000:00FF8B4A   25 FF 00 00 00                      AND EAX,000000FFH           ;ELSE CLEAR ALL  BUT AL
0000:00FF8B4F   2C 10                               SUB AL,10H
0000:00FF8B51   B1 02                               MOV CL,02H
0000:00FF8B53   D2 E0                               SHL AL,CL                   ;GET ADDRESSOF ROUTINE TO
0000:00FF8B55   03 F0                               ADD ESI,EAX                 ;SERVICE  THE FUNCTION KEY
0000:00FF8B57   8B 3E                               MOV  EDI,[ESI]              ;THAT WAS  PRESSED
0000:00FF8B59   FF D7                               CALL EDI                    ;CALL THE ROUTINE
0000:00FF8B5B   EB AC                               JMP PRE_COM                 ;GET NEXT  KEY PRESS
0000:00FF8B5D
0000:00FF8B5D   E8 8C 00 00 00      CHANGE_DATA:    CALL DATA_INPUT             ;CHANGE DATA AT CURRENT ADDRESS
0000:00FF8B62   EB A5                               JMP PRE_COM                     ;GET  NEXT KEY PRESS
0000:00FF8B64   E8 13 01 00 00      CHANGE_ADDRESS: CALL  ADDRESS                   ;CHANGE CURRENT ADDRESS
0000:00FF8B69   EB AE                               JMP KEY_INTR                ;SERVICE NEXT KEY
0000:00FF8B6B   E8 BA 01 00 00      AUTO_CHANGE:    CALL AUTO                   ;PUT IN AUTO LOAD MODE
0000:00FF8B70   EB A7                               JMP KEY_INTR                ;SERVICE NEXT KEY
0000:00FF8B72   A1 44 02 00 00      SINGLE_STEP:    MOV EAX,USER_EFLAGS         ;GET EFLAGS FROM USER TSS
0000:00FF8B77   0D 00 01 00 00                      OR EAX,00000100H            ;SET TF FLAG
```



```
0000:00FF8B7C A3 44 02 00 00    RUN_USER: MOV USER_EFLAGS,EAX
0000:00FF8B81 B8 00 00 20 00    RUN_USER_A:  MOV EAX,200000H
0000:00FF8B86 A3 B0 00 00 00            MOV INVALID_TSS,EAX
0000:00FF8B8B B8 00 85 00 00            MOV EAX,08500H

0000:00FF8B90 A3 B4 00 00 00                    MOV USE_EFLAGS,EAX              ;WRITE IT TO USER TSS
0000:00FF8B95 BE 37 84 FF 00                    MOV ESI,OFFSET BLANK            ;CLEAR DISPLAY
0000:00FF8B9A BF EC 02 00 00                    MOV EDI,OFFSET DISPBF
0000:00FF8B9F B9 11 00 00 00                    MOV ECX,17
0000:00FF8BA4 3E F3 A4                          REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF8BA7 E8 BB 0B 00 00                    CALL LCD_DISPLAY
0000:00FF8BAC A1 44 02 00 00                    MOV EAX,USER_EFLAGS             ;SET RESUME FLAG OF
0000:00FF8BB1 0D 00 00 01 00                    OR EAX,00010000H                ;USER_EFLAGS
0000:00FF8BB6 A3 44 02 00 00                    MOV USER_EFLAGS,EAX
0000:00FF8BBB
0000:00FF8BBB CF                                IRET                            ;RETURN TO USER TASK
0000:00FF8BBC
                                        ;-----------------------------------------------------------------------
                                            ;THIS  POINT IS REACHED WHEN A NON-MASKABLE INTERRUPT OR A DEBUG
                                            ;EXCEPTION IS GENERATED WHILE A USER PROGRAM IS  RUNNING
                                        ;-----------------------------------------------------------------------
0000:00FF8BBC
0000:00FF8BBC A1 40 02 00 00                    MOV EAX,USER_EIP                ;LOAD THE CURRENT ADDRESS
0000:00FF8BC1 A3 A0 02 00 00                    MOV CURRENT_ADDRESS,EAX         ;WITH THE USER EIP
0000:00FF8BC6 BF F4 02 00 00                    MOV EDI,OFFSET DISPBF+8         ;LOAD ARROW  INTO DISPLAY
0000:00FF8BCB C6 07 7E                          MOV [EDI],BYTE PTR 7EH
0000:00FF8BCE E8 AE 0D 00 00                    CALL DISPLAY_INFO               ;DISPLAY ADDRESS AND DATA
0000:00FF8BD3 33 C0                             XOR EAX,EAX                     ;CLEAD DB6 DEBUG REGISTER
0000:00FF8BD5 0F 23 F0                          MOV DR6,EAX
0000:00FF8BD8 A1 44 02 00 00                    MOV EAX,USER_EFLAGS             ;CLEAR TF FLAG OF USER
0000:00FF8BDD 25 FF FE FF FF                    AND EAX,FFFFFEFFH               ;TASK
0000:00FF8BE2 A3 44 02 00 00                    MOV USER_EFLAGS,EAX
0000:00FF8BE7 E9 1D FF FF FF                    JMP PRE_COM                     ;GET NEXT KEY PRESS
                                        ;-----------------------------------------------------------------------
0000:00FF8BEC 90                        NULL_ROUTINE:   NOP
0000:00FF8BED C3                                RET
                                        ;-----------------------------------------------------------------------
                                          ;THE ROUTINE  DATA_INPUT SHIFTS THE DATA AT THE CURRENT ADDRESS TO THE
                                          ;LEFT BY 4 BITS, INSERTS  THE LOWER 4 BITS OF AL INTO THE LOWER 4 BITS OF
                                          ;THE DATA, AND DISPLAYS THE NEW DATA AND THE CURRENT ADDRESS.

0000:00FF8BEE
0000:00FF8BEE 53                        DATA_INPUT:     PUSH EBX
0000:00FF8BEF 51                                PUSH ECX
0000:00FF8BF0 56                                PUSH ESI
0000:00FF8BF1
0000:00FF8BF1 BE F4 02 00 00                    MOV ESI,OFFSET DISPBF+8         ;INSERT ARROW  IN DISPLAY
0000:00FF8BF6 C6 06 7E                          MOV [ESI],BYTE PTR 7EH          ;TO POINT TO DATAFIELD
0000:00FF8BF9 8B 35 A0 02 00 00                 MOV ESI,CURRENT_ADDRESS         ;GET CURRENT ADDRESS
0000:00FF8BFF 8B 0D 1C 02 00 00                 MOV ECX,A_MODE                  ;GET DATA MODE
0000:00FF8C05 BB 00 80 00 00                    MOV EBX,00008000H               ;GET LAST RAM BYTE+1
0000:00FF8C0A 2B D9                             SUB EBX,ECX                     ;FIND LAST ADDRESS OF RAM
0000:00FF8C0C 3B F3                             CMP ESI,EBX                     ;FOR CURRENT DATA  MODE
0000:00FF8C0E 0F 8E 0A 00 00 00                 JLE GET_DATA                ;IF LESS THAN OR EQUAL CONTINUE
0000:00FF8C14 E8 68 0D 00 00                    CALL DISPLAY_INFO               ;ELSE  DISPLAY xxxxxxxx
0000:00FF8C19 E9 4E 00 00 00                    JMP END_DATA_INPUT              ;AND RETURN
0000:00FF8C1E
0000:00FF8C1E 33 DB                     GET_DATA:       XOR EBX,EBX                     ;CLEAD EBX
0000:00FF8C20 83 F9 04                          CMP ECX,00000004H               ;IF DATA MODE=4
0000:00FF8C23 0F 84 1C 00 00 00                 JZ GET_LWORD                    ;GET LONG WORD
0000:00FF8C29 83 F9 02                          CMP ECX,00000002H               ;IF DATA MODE=2
0000:00FF8C2C 0F 84 26 00 00 00                 JZ GET_WORD                     ;GET  WORD
0000:00FF8C32 8A 1E                             MOV BL,[ESI]            ;ELSE GET DATA BYTE  AT CURRENT ADDRESS
0000:00FF8C34 E8 37 00 00 00                    CALL FIX_DATA                   ;UPDATE DATA
0000:00FF8C39 88 1E                             MOV [ESI],BL                ;WRITE IT TO CURRENT ADDRESS
0000:00FF8C3B E8 41 0D 00 00                    CALL DISPLAY_INFO               ;DISPLAY ADDRESS AND  DATA
0000:00FF8C40 E9 27 00 00 00                    JMP END_DATA_INPUT
0000:00FF8C45
0000:00FF8C45 8B 1E                     GET_LWORD:      MOV EBX,[ESI]                   ;GET  LONG WORD
0000:00FF8C47 E8 24 00 00 00                    CALL FIX_DATA                   ;UPDATE DATA
0000:00FF8C4C 89 1E                             MOV [ESI],EBX                   ;WRITE IT TO CURRENT ADDRESS
0000:00FF8C4E E8 2E 0D 00 00                    CALL DISPLAY_INFO               ;DISPLAY DATA AND ADDRESS
0000:00FF8C53 E9 14 00 00 00                    JMP END_DATA_INPUT
0000:00FF8C58
0000:00FF8C58 8A 1E                     GET_WORD:       MOV BL,[ESI]                    ;GET  LOW  BYTE OF DATA
0000:00FF8C5A 46                                INC ESI
0000:00FF8C5B 8A 3E                             MOV BH,[ESI]                    ;GET HIGH BYTE OF DATA
0000:00FF8C5D E8 0E 00 00 00                    CALL FIX_DATA                   ;UPDATE DATA
0000:00FF8C62 88 3E                             MOV [ESI],BH                    ;WRITE HIGH BYTE OF DATA
0000:00FF8C64 4E                                DEC ESI                         ;TO CURRENT ADDRESS
0000:00FF8C65 88 1E                             MOV [ESI],BL                    ;WRITE LOW BYTE OF DATA
0000:00FF8C67 E8 15 0D 00 00                    CALL DISPLAY_INFO               ;DISPLAY DATA AND ADDRESS
0000:00FF8C6C
0000:00FF8C6C 5E                        END_DATA_INPUT: POP ESI
0000:00FF8C6D 59                                POP ECX
0000:00FF8C6E 5B                                POP EBX
0000:00FF8C6F C3                                RET
0000:00FF8C70
0000:00FF8C70 C1 C3 04                  FIX_DATA:       ROL EBX,04H                 ;SHIFT DATA 4 BITS
0000:00FF8C73 81 E3 F0 FF FF FF                 AND EBX,FFFFFFF0H               ;CLEAR LOW 4 BITS
0000:00FF8C79 0A D8                             OR BL,AL                        ;INSERT NEW DATA FROM KEY PRESS
0000:00FF8C7B C3                                RET
0000:00FF8C7C
                                        ;-----------------------------------------------------------------------
                                        ;ADDRESS - SERVICES THE  ADDR KEY. THIS ROUTINE IS CALLED WHEN THE ADDR KEY
                                        ;IS PRESSED AND RETAINS CONTROL  UNTIL A NON-HEX KEY IS PRESSED. WITH EACH
                                        ;HEX KEY PRESSED, THE CURRENT ADDRESS IS SHIFTED LEFT FOUR BITS AND THE VALUE
                                        ;OF THE HEX KEY IS INSERTED INTO THE LOW ORDER 4 BITS OF THE CURRENT ADDRESS
                                        ;THE DISPLAY IS UPDATED WITH EACH HEX KEY PRESS.
```

```
1228                                    ;-------------------------------------------------------------------------
1229    0000:00FF8C7C
1230    0000:00FF8C7C   53              ADDRESS:        PUSH EBX
1231    0000:00FF8C7D   56                              PUSH ESI
1232    0000:00FF8C7E
1233    0000:00FF8C7E   BE F4 02 00 00                  MOV ESI,OFFSET DISPBF+8     ;FIX  ARROW  IN DISPLAY  TO
1234    0000:00FF8C83   C6 06 7F                        MOV [ESI],BYTE PTR 7FH      ;POINT TO  ADDRES FIELD
1235    0000:00FF8C86   E8 F6 0C 00 00  NEXT_ADDR_INPUT:CALL DISPLAY_INFO           ;DISPLAY CURRENT ADDRESS & DATA
1236    0000:00FF8C8B   E8 E4 0B 00 00                  CALL KEYBOARD               ;GET NEXT KEY PRESS
1237    0000:00FF8C90   3C 0F                           CMP AL,0FH                  ;WAS HEXKEY PRESSED?
1238    0000:00FF8C92   0F 8F 19 00 00 00               JG NO_ADDRESS               ;IF GO BACK TO MAIN LOOP
1239    0000:00FF8C98   BE A0 02 00 00                  MOV ESI,OFFSET CURRENT_ADDRESS          ;GET CURRENT ADDRESS
1240    0000:00FF8C9D   8B 1E                           MOV EBX,[ESI]
1241    0000:00FF8C9F   C1 C3 04                        ROL EBX,04H                             ;SHIFT LEFT 4  BITS
1242    0000:00FF8CA2   80 E3 F0                        AND BL,F0H                              ;CLEAR LOW 4 BITS
1243    0000:00FF8CA5   0A D8                           OR BL,AL                    ;INSERT NEW DATA INTO ADDRESS
1244    0000:00FF8CA7   81 E3 FF FF FF 00               AND EBX,00FFFFFFH           ;CLEAR TOP BYTE
1245    0000:00FF8CAD   89 1E                           MOV [ESI],EBX               ;WR UPDTD ADDR TO CURRNT_ADDR
1246
1247    0000:00FF8CAF   EB D5                           JMP NEXT_ADDR_INPUT                     ;BACK  TO MAIN LOOP
1248    0000:00FF8CB1
1249    0000:00FF8CB1   5E              NO_ADDRESS:     POP ESI
1250    0000:00FF8CB2   5B                              POP EBX
1251    0000:00FF8CB3   C3                              RET
1252                                    ;-------------------------------------------------------------------------
1253                                    ;THESE ROUTINES  CHANGE THE DATA  ENTRY MODE TO BYTE, WORD OR LONG WORD
1254                                    ;THE DISPLAY IS UPDATED CORRESPONDINGLY
1255                                    ;-------------------------------------------------------------------------
1256    0000:00FF8CB4
1257    0000:00FF8CB4   B8 01 00 00 00  BYTE_MODE:      MOV EAX,00000001H                       ;SET TO BYTE MODE
1258    0000:00FF8CB9   E9 0F 00 00 00                  JMP MODE
1259    0000:00FF8CBE
1260    0000:00FF8CBE   B8 02 00 00 00  WORD_MODE:      MOV EAX,00000002H                       ;SET TO WORD MODE
1261    0000:00FF8CC3   E9 05 00 00 00                  JMP MODE
1262    0000:00FF8CC8
1263    0000:00FF8CC8   B8 04 00 00 00  LWORD_MODE:     MOV EAX,00000004H                       ;SET TO LWORD MODE
1264    0000:00FF8CCD
1265    0000:00FF8CCD   BE F4 02 00 00  MODE:           MOV ESI,OFFSET DISPBF+8         ;DISPLAY ARROW POINTS  TO
1266    0000:00FF8CD2   C6 06 7E                        MOV [ESI],BYTE PTR 7EH          ;DATA FIELD
1267    0000:00FF8CD5   A3 1C 02 00 00                  MOV A_MODE,EAX                  ;UPDATE DATA_MODE
1268    0000:00FF8CDA   E8 A2 0C 00 00                  CALL DISPLAY_INFO               ;DISPLAY DATA & ADDRESS
1269    0000:00FF8CDF   C3                              RET
1270                                    ;-------------------------------------------------------------------------
1271                                    ;THIS ROUTINE INCREMENTS THE  CURRENT ADDRESS BY ONE BYTE, WORD OR LONG
1272                                    ;WORD, DEPENDING ON THE CURRENT  DATA MODE AND UPDATES THE DISPLAY
1273                                    ;-------------------------------------------------------------------------
1274    0000:00FF8CE0
1275    0000:00FF8CE0   A1 A0 02 00 00  INCREMENT:      MOV EAX,CURRENT_ADDRESS         ;GET CURRENT ADDRESS
1276    0000:00FF8CE5   8B 1D 1C 02 00 00               MOV EBX,A_MODE                  ;GET CURRENT DATA  MODE
1277    0000:00FF8CEB   03 C3                           ADD EAX,EBX                     ;ADD  IT TO CURRENT ADDRESS
1278    0000:00FF8CED   25 FF FF FF 00                  AND EAX,00FFFFFFH               ;CLEAR  HIGH BYTE
1279    0000:00FF8CF2   A3 A0 02 00 00                  MOV CURRENT_ADDRESS,EAX         ;UPDATE  CURRENT ADDRESS
1280    0000:00FF8CF7   BE F4 02 00 00                  MOV ESI,OFFSET DISPBF+8         ;SET DISPLAY ARROW TO POINT
1281    0000:00FF8CFC   C6 06 7E                        MOV [ESI],BYTE PTR 7EH          ;TO DATA FIELD
1282    0000:00FF8CFF   E8 7D 0C 00 00                  CALL DISPLAY_INFO               ;UPDATE DISPLAY
1283    0000:00FF8D04   C3                              RET
1284    0000:00FF8D05
1285                                    ;-------------------------------------------------------------------------
1286                                    ;THIS ROUTINE DECREMENTS THE  CURRENT ADDRESS BY ONE BYTE, WORD OR LONG
1287                                    ;WORD, DEPENDING ON THE CURRENT  DATA MODE AND UPDATES THE DISPLAY
1288                                    ;-------------------------------------------------------------------------
1289    0000:00FF8D05
1290    0000:00FF8D05   A1 A0 02 00 00  DECREMENT:      MOV EAX,CURRENT_ADDRESS         ;GET  CURRENT ADDRESS
1291    0000:00FF8D0A   8B 1D 1C 02 00 00               MOV EBX,A_MODE                  ;GET CURRENT DATA MODE
1292    0000:00FF8D10   2B C3                           SUB EAX,EBX                     ;SUB IT FROM CURRENT ADDRESS
1293    0000:00FF8D12   25 FF FF FF 00                  AND EAX,00FFFFFFH               ;CLEAR HIGH BYTE
1294    0000:00FF8D17   A3 A0 02 00 00                  MOV CURRENT_ADDRESS,EAX         ;UPDATE  CURRENT ADDRESS
1295    0000:00FF8D1C   BE F4 02 00 00                  MOV ESI,OFFSET DISPBF+8         ;SET DISPLAY ARROW  TO POINT
1296    0000:00FF8D21   C6 06 7E                        MOV [ESI],BYTE PTR 7EH          ;TO DATA FIELD
1297    0000:00FF8D24   E8 58 0C 00 00                  CALL DISPLAY_INFO               ;UPDATE DISPLAY
1298    0000:00FF8D29   C3                              RET
1299    0000:00FF8D2A
1300                                    ;-------------------------------------------------------------------------
1301                                    ;AUTO - SREVICES THE AUTO KEY. THIS ROUTINE AUTOMATICALLY INCREMENTS  THE
1302                                    ;CURRENT ADDRESS AFTER THE DATA AT THE CURRENT  ADDRESS HAS BEEN MODIFIED.
1303                                    ;THIS ROUTINE RETAINS  CONTROL UNTIL A NON-HEX KEY IS PRESSED
1304                                    ;-------------------------------------------------------------------------
1305    0000:00FF8D2A
1306    0000:00FF8D2A   BE F4 02 00 00  AUTO:           MOV ESI,OFFSET DISPBF+8         ;SET DISPLAY ARROW  TO POINT
1307    0000:00FF8D2F   C6 06 7E                        MOV [ESI],BYTE PTR 7EH          ;TO POINT TO DATA
1308    0000:00FF8D32   8B 0D 1C 02 00 00  NEXT_ADDRESS: MOV ECX,A_MODE                 ;GET DATA MODE
1309    0000:00FF8D38   D1 C1                           ROL ECX                         ;MULTIPLY BY 2 TO GET #
1310                                                                                    ;OF HEX  DIGITS/ADDRESS
1311    0000:00FF8D3A   E8 42 0C 00 00                  CALL DISPLAY_INFO
1312    0000:00FF8D3F   E8 30 0B 00 00  NEXT_HEX_KEY:   CALL KEYBOARD                  ;GET NEXT KEY
1313    0000:00FF8D44   3C 0F                           CMP AL,0FH                      ;WAS HEX KEY PRESSED?
1314    0000:00FF8D46   0F 8F 0E 00 00 00               JG END_AUTO                     ;IF NOT  RETURN
1315    0000:00FF8D4C   E8 9D FE FF FF                  CALL DATA_INPUT                 ;ELSE CHANGE DATA
1316    0000:00FF8D51   E2 EC                           LOOP NEXT_HEX_KEY               ;GET NEXT DATA INPUT
1317    0000:00FF8D53   E8 88 FF FF FF                  CALL INCREMENT                  ;INCREMENT CURRENT ADDRESS
1318    0000:00FF8D58   EB D8                           JMP NEXT_ADDRESS                ;START AGAIN
1319    0000:00FF8D5A
1320    0000:00FF8D5A   C3              END_AUTO:       RET
1321    0000:00FF8D5B
1322                                    ;-------------------------------------------------------------------------
1323    0000:00FF8D5B                           ORG 00FF8D5BH
1324    0000:00FF8D5B   E8 AA 0C 00 00  MOVE_DATA:CALL SUB_1
1325    0000:00FF8D60   3C 13                   CMP AL,13H
```



```
1326  0000:00FF8D62  0F 84 95 00 00 00          JZ M_D_6
1327  0000:00FF8D68  A1 A8 02 00 00             MOV EAX,START_ADDRESS
1328  0000:00FF8D6D  3D 00 03 00 00             CMP EAX,0300H
1329  0000:00FF8D72  0F 8C 8B 00 00 00          JL M_D_2
1330  0000:00FF8D78  3B 05 9C 00 00 00          CMP EAX,[MEM_VAL]
1331  0000:00FF8D7E  0F 8F 7F 00 00 00          JG M_D_2
1332  0000:00FF8D84  8B 1D AC 02 00 00          MOV EBX,END_ADDRESS
1333  0000:00FF8D8A  81 FB 00 03 00 00          CMP EBX,0300H
1334  0000:00FF8D90  0F 8C 6D 00 00 00          JL M_D_2
1335  0000:00FF8D96  3B 0D 9C 00 00 00          CMP ECX,[MEM_VAL]
1336  0000:00FF8D9C  0F 8F 61 00 00 00          JG M_D_2
1337  0000:00FF8DA2  3B C3                      CMP EAX,EBX
1338  0000:00FF8DA4  0F 8F 63 00 00 00          JG M_D_4
1339  0000:00FF8DAA  8B 15 E8 02 00 00          MOV EDX,MOVE_INFO
1340  0000:00FF8DB0  3B C2                      CMP EAX,EDX
1341  0000:00FF8DB2  0F 8F 1F 00 00 00          JG M_D_1
1342  0000:00FF8DB8  8B FA                      MOV EDI,EDX
1343  0000:00FF8DBA  8B F0                      MOV ESI,EAX
1344  0000:00FF8DBC  8B CB                      MOV ECX,EBX
1345  0000:00FF8DBE  2B C8                      SUB ECX,EAX
1346  0000:00FF8DC0  03 D1                      ADD EDX,ECX
1347  0000:00FF8DC2  3B 15 9C 00 00 00          CMP EDX,[09CH]
1348  0000:00FF8DC8  0F 8F 35 00 00 00          JG M_D_2
1349  0000:00FF8DCE  41                         INC ECX
1350  0000:00FF8DCF  3E F3 A4                   REP MOVSB DS:[EDI],CS:[ESI]
1351  0000:00FF8DD2  E9 1C 00 00 00             JMP M_D_7
1352  0000:00FF8DD7  81 FA 00 03 00 00  M_D_1:  CMP EDX,0300H
1353  0000:00FF8DDD  0F 8C 20 00 00 00          JL M_D_2
1354  0000:00FF8DE3  8B F3                      MOV ESI,EBX
1355  0000:00FF8DE5  8B CB                      MOV ECX,EBX
1356  0000:00FF8DE7  2B C8                      SUB ECX,EAX
1357  0000:00FF8DE9  8B FA                      MOV EDI,EDX
1358  0000:00FF8DEB  03 F9                      ADD EDI,ECX
1359  0000:00FF8DED  41                         INC ECX
1360  0000:00FF8DEE  FD                         STD
1361  0000:00FF8DEF  3E F3 A4                   REP MOVSB DS:[EDI],CS:[ESI]
1362  0000:00FF8DF2  FC                 M_D_5:  CLD
1363  0000:00FF8DF3  A1 E8 02 00 00     M_D_7:  MOV EAX,MOVE_INFO
1364  0000:00FF8DF8  A3 A0 02 00 00             MOV CURRENT_ADDRESS,EAX
1365  0000:00FF8DFD  E8 7F 0B 00 00     M_D_6:  CALL DISPLAY_INFO
1366  0000:00FF8E02  C3                         RET
1367  0000:00FF8E03  BE 8A 84 FF 00     M_D_2:  MOV ESI,OFFSET MESSAGE_1
1368  0000:00FF8E08  E9 05 00 00 00             JMP M_D_3
1369  0000:00FF8E0D  BE 9B 84 FF 00     M_D_4:  MOV ESI,OFFSET MESSAGE_2
1370  0000:00FF8E12  BF EC 02 00 00     M_D_3:  MOV EDI, OFFSET DISPBF
1371  0000:00FF8E17  B9 11 00 00 00             MOV ECX,00000011H
1372  0000:00FF8E1C  3E F3 A4                   REP MOVSB DS:[EDI],CS:[ESI]
1373  0000:00FF8E1F  C3                         RET
1374  0000:00FF8E20  E8 91 01 00 00     UP_LOAD:CALL SUB_2
1375  0000:00FF8E25  3C FF                      CMP AL,FFH
1376  0000:00FF8E27  0F 84 65 00 00 00          JZ U_L_1
1377  0000:00FF8E2D  A1 A4 02 00 00             MOV EAX,DATA_MODE
1378  0000:00FF8E32  3D 0A 00 00 00             CMP EAX,0AH
1379  0000:00FF8E37  0F 84 0A 00 00 00          JZ U_L_2
1380  0000:00FF8E3D  BF 7D 8E FF 00             MOV EDI,OFFSET U_L_3
1381  0000:00FF8E42  E9 05 00 00 00             JMP U_L_4
1382  0000:00FF8E47  BF 6A 8E FF 00     U_L_2:  MOV EDI,OFFSET U_L_5
1383  0000:00FF8E4C  B0 06              U_L_4:  MOV AL,06H
1384  0000:00FF8E4E  52                         PUSH EDX
1385  0000:00FF8E4F  C7 C2 08 F0 00 00          MOV EDX,S_PORT_8
1386  0000:00FF8E55  EE                         OUT DX,AL
1387  0000:00FF8E56  5A                         POP EDX
1388  0000:00FF8E57  8B 35 A8 02 00 00          MOV ESI,START_ADDRESS
1389  0000:00FF8E5D  BB 93 8E FF 00             MOV EBX,OFFSET U_L_6
1390  0000:00FF8E62  33 CB                      XOR ECX,EBX
1391  0000:00FF8E64  33 C0                      XOR EAX,EAX
1392  0000:00FF8E66  8A 06                      MOV AL,[ESI]
1393  0000:00FF8E68  FF E7                      JMP EDI
1394  0000:00FF8E6A  8A E0              U_L_5:  MOV AH,AL
1395  0000:00FF8E6C  B1 04                      MOV CL,04H
1396  0000:00FF8E6E  D2 C8                      ROR AL,CL
1397  0000:00FF8E70  24 0F                      AND AL,0FH
1398  0000:00FF8E72  D7                         XLAT
1399  0000:00FF8E73  E8 2B 00 00 00             CALL U_L_7
1400  0000:00FF8E78  8A C4                      MOV AL,AH
1401  0000:00FF8E7A  24 0F                      AND AL,0FH
1402  0000:00FF8E7C  D7                         XLAT
1403  0000:00FF8E7D  E8 21 00 00 00     U_L_3:  CALL U_L_7
1404  0000:00FF8E82  46                         INC ESI
1405  0000:00FF8E83  4A                         DEC EDX
1406  0000:00FF8E84  75 E0                      JNE E0H
1407  0000:00FF8E86  B9 FF FF 00 00             MOV ECX,FFFFH
1408  0000:00FF8E8B  E2 FE                      LOOP FEH
1409  0000:00FF8E8D  E9 9D 00 00 00             JMP D_L_10
1410  0000:00FF8E92  C3                 U_L_1:  RET
1411  0000:00FF8E93  30 31 32 33 34 35  U_L_6:  DB 30H,31H,32H,33H,34H,35H
1412  0000:00FF8E99  36 37 38 39                DB 36H,37H,38H,39H
1413  0000:00FF8E9D  41 42 43 44 45 46          DB 41H,42H,43H,44H,45H,46H
1414  0000:00FF8EA3  52                 U_L_7:  PUSH EDX
1415  0000:00FF8EA4  50                         PUSH EAX
1416  0000:00FF8EA5  BA 04 F0 00 00             MOV EDX,OFFSET S_PORT_4
1417  0000:00FF8EAA  EC                 U_L_8:  IN AL,DX
1418  0000:00FF8EAB  24 04                      AND AL,04H
1419  0000:00FF8EAD  74 FB                      JE U_L_8
1420  0000:00FF8EAF  58                         POP EAX
1421  0000:00FF8EB0  BA 0C F0 00 00             MOV EDX,OFFSET S_PORT_C
1422  0000:00FF8EB5  EE                         OUT DX,AL
1423  0000:00FF8EB6  5A                         POP EDX
```

```
1424  0000:00FF8EB7  C3                          RET
1425  0000:00FF8EB8  E8 F9 00 00 00     DN_LOAD:  CALL SUB_2
1426  0000:00FF8EBD  3C FF                       CMP AL,FFH
1427  0000:00FF8EBF  74 D1                       JZ U_L_1
1428  0000:00FF8EC1  A1 A4 02 00 00              MOV EAX,DATA_MODE
1429  0000:00FF8EC6  3D 0A 00 00 00              CMP EAX,0AH
1430  0000:00FF8ECB  0F 84 0A 00 00 00           JZ X_1
1431  0000:00FF8ED1  BF 29 8F FF 00              MOV EDI,0FF8F29H
1432  0000:00FF8ED6  E9 05 00 00 00              JMP X_2
1433  0000:00FF8EDB  BF 0A 8F FF 00     X_1:     MOV EDI,0FF8F0AH
1434  0000:00FF8EE0  B0 09              X_2:     MOV AL,09H
1435  0000:00FF8EE2  52                          PUSH EDX
1436  0000:00FF8EE3  C7 C2 08 F0 00 00           MOV EDX,S_PORT_8
1437  0000:00FF8EE9  EE                          OUT DX,AL
1438  0000:00FF8EEA  5A                          POP EDX
1439  0000:00FF8EEB  8B 35 A8 02 00 00           MOV ESI,START_ADDRESS
1440  0000:00FF8EF1  BB 9F 8F FF 00              MOV EBX,0FF8F9FH
1441  0000:00FF8EF6  33 C9                       XOR ECX,ECX
1442  0000:00FF8EF8  33 C0                       XOR EAX,EAX
1443  0000:00FF8EFA  E8 47 00 00 00              CALL D_L_9
1444  0000:00FF8EFF  FF E7                       JMP EDI
1445  0000:00FF8F01  B4 00                       MOV AH,00H
1446  0000:00FF8F03  E8 53 00 00 00              CALL SUB_3
1447  0000:00FF8F08  FF E7                       JMP EDI
1448  0000:00FF8F0A  E8 7B 00 00 00              CALL SUB_4
1449  0000:00FF8F0F  3C FF                       CMP AL,FFH
1450  0000:00FF8F11  74 EE                       JZ EEH
1451  0000:00FF8F13  8A E0                       MOV AH,AL
1452  0000:00FF8F15  B1 04                       MOV CL,04H
1453  0000:00FF8F17  D2 C4                       ROL AH,CL
1454  0000:00FF8F19  E8 3D 00 00 00              CALL SUB_3
1455  0000:00FF8F1E  E8 67 00 00 00              CALL SUB_4
1456  0000:00FF8F23  3C FF                       CMP AL,FFH
1457  0000:00FF8F25  74 F2                       JZ F2H
1458  0000:00FF8F27  0A C4              D_L_8:   OR AL,AH
1459  0000:00FF8F29  88 06                       MOV [ESI],AL
1460  0000:00FF8F2B  46                          INC ESI
1461  0000:00FF8F2C  4A                          DEC EDX
1462  0000:00FF8F2D  75 D2                       JNE D2H
1463  0000:00FF8F2F  B0 0A              D_L_10:   MOV AL,0AH
1464  0000:00FF8F31  C7 C2 08 F0 00 00           MOV EDX,S_PORT_8
1465  0000:00FF8F37  EE                          OUT DX,AL
1466  0000:00FF8F38  4E                          DEC ESI
1467  0000:00FF8F39  8B C6                       MOV EAX,ESI
1468  0000:00FF8F3B  A3 A0 02 00 00              MOV 002A0H,EAX
1469  0000:00FF8F40  E8 3C 0A 00 00              CALL 0A3CH
1470  0000:00FF8F45  C3                          RET
1471  0000:00FF8F46  52                 D_L_9:   PUSH EDX
1472  0000:00FF8F47  C7 C2 04 F0 00 00           MOV EDX,S_PORT_4
1473  0000:00FF8F4D  EC                          IN AL,DX
1474  0000:00FF8F4E  24 01                       AND AL,01H
1475  0000:00FF8F50  74 FB                       JZ FBH
1476  0000:00FF8F52  C7 C2 0C F0 00 00           MOV EDX,S_PORT_C
1477  0000:00FF8F58  EC                          IN AL,DX
1478  0000:00FF8F59  5A                          POP EDX
1479  0000:00FF8F5A  C3                          RET
1480  0000:00FF8F5B  51                 SUB_3:   PUSH ECX
1481  0000:00FF8F5C  52                          PUSH EDX
1482  0000:00FF8F5D  B9 FF FF 00 00              MOV ECX,FFFFH
1483  0000:00FF8F62  C7 C2 04 F0 00 00           MOV EDX,S_PORT_4
1484  0000:00FF8F68  49                          DEC ECX
1485  0000:00FF8F69  0F 84 0F 00 00 00           JZ D_L_5
1486  0000:00FF8F6F  EC                          IN AL,DX
1487  0000:00FF8F70  24 01                       AND AL,01H
1488  0000:00FF8F72  74 F4                       JE F4H
1489  0000:00FF8F74  C7 C2 0C F0 00 00           MOV EDX,S_PORT_C
1490  0000:00FF8F7A  EC                          IN AL,DX
1491  0000:00FF8F7B  5A                          POP EDX
1492  0000:00FF8F7C  59                          POP ECX
1493  0000:00FF8F7D  C3                          RET
1494  0000:00FF8F7E  5A                 D_L_5:   POP EDX
1495  0000:00FF8F7F  59                          POP ECX
1496  0000:00FF8F80  5B                 D_L_1:   POP EBX
1497  0000:00FF8F81  B0 00                       MOV AL,00H
1498  0000:00FF8F83  BA 01 00 00 00              MOV EDX,OFFSET 01H
1499  0000:00FF8F88  EB 9D                       JMP D_L_8
1500  0000:00FF8F8A  2C 30              SUB_4:   SUB AL,30H
1501  0000:00FF8F8C  0F 8C 0A 00 00 00           JL D_L_2
1502  0000:00FF8F92  3C 16                       CMP AL,16H
1503  0000:00FF8F94  0F 8F 02 00 00 00           JG D_L_2
1504  0000:00FF8F9A  D7                          XLAT
1505  0000:00FF8F9B  C3                          RET
1506  0000:00FF8F9C  B0 FF              D_L_2:   MOV AL,FFH
1507  0000:00FF8F9E  C3                          RET
1508  0000:00FF8F9F  00 01 02 03 04 05           DB 00H,01H,02H,03H,04H,05H
1509  0000:00FF8FA5  06 07 08 09                 DB 06H,07H,08H,09H
1510  0000:00FF8FA9  FF FF FF FF FF              DB FFH,FFH,FFH,FFH,FFH,FFH,FFH
      0000:00FF8FAF  FF
1511  0000:00FF8FB0  0A 0B 0C 0D 0E 0F           DB 0AH,0BH,0CH,0DH,0EH,0FH
1512  0000:00FF8FB6  E8 6F 0A 00 00     SUB_2:   CALL 0A6FH
1513  0000:00FF8FBB  3C 13                       CMP AL,13H
1514  0000:00FF8FBD  0F 84 91 00 00 00           JZ D_L_3
1515  0000:00FF8FC3  A1 A4 02 00 00              MOV EAX,DATA_MODE
1516  0000:00FF8FC8  3D 0A 00 00 00              CMP EAX,0AH
1517  0000:00FF8FCD  0F 84 07 00 00 00           JZ D_L_4
1518  0000:00FF8FD3  3D 0B 00 00 00              CMP EAX,0BH
1519  0000:00FF8FD8  75 DC                       JNE DCH
1520  0000:00FF8FDA  E8 66 0A 00 00     D_L_4:   CALL OFFSET 0A66H
```



```
1521  0000:00FF8FDF  3C 13                        CMP AL,13H
1522  0000:00FF8FE1  74 D3                        JE D3H
1523  0000:00FF8FE3  E8 78 0A 00 00               CALL OFFSET 0A78H
1524  0000:00FF8FE8  3C 13                        CMP AL,13H
1525  0000:00FF8FEA  74 EE                        JE EEH
1526  0000:00FF8FEC  A1 A8 02 00 00               MOV EAX,START_ADDRESS
1527  0000:00FF8FF1  3D 00 03 00 00               CMP EAX,0300H
1528  0000:00FF8FF6  0F 8C 39 00 00 00            JL D_L_6
1529  0000:00FF8FFC  3B 05 9C 00 00 00            CMP EAX,[MEM_VAL]
1530  0000:00FF9002  0F 8F 2D 00 00 00            JG D_L_6
1531  0000:00FF9008  8B 15 AC 02 00 00            MOV EDX,END_ADDRESS
1532  0000:00FF900E  81 FA 00 03 00 00            CMP EDX,0300H
1533  0000:00FF9014  0F 8C 1B 00 00 00            JL D_L_6
1534  0000:00FF901A  3B 15 9C 00 00 00            CMP EDX,MEM_VAL
1535  0000:00FF9020  0F 8F 0F 00 00 00            JG D_L_6
1536  0000:00FF9026  2B D0                        SUB EDX,EAX
1537  0000:00FF9028  0F 8C 11 00 00 00            JL D_L_7
1538  0000:00FF902E  42                           INC EDX
1539  0000:00FF902F  E8 28 00 00 00               CALL 28H
1540  0000:00FF9034  C3                           RET
1541  0000:00FF9035  BE 8A 84 FF 00     D_L_6:   MOV ESI,OFFSET 0FF848AH
1542  0000:00FF903A  E9 05 00 00 00               JMP 05H
1543  0000:00FF903F  BE 9B 84 FF 00     D_L_7:   MOV ESI,0FF849BH
1544  0000:00FF9044  B0 FF                        MOV AL,FFH
1545  0000:00FF9046  BF EC 02 00 00               MOV EDI,OFFSET 02ECH
1546  0000:00FF904B  B9 11 00 00 00               MOV ECX,OFFSET 11H
1547  0000:00FF9050  3E F3 A4                     REP MOVSB DS:[EDI],CS:[ESI]
1548  0000:00FF9053  C3                           RET
1549  0000:00FF9054  B0 FF              D_L_3:    MOV AL,FFH
1550  0000:00FF9056  E8 26 09 00 00               CALL 0926H
1551  0000:00FF905B  C3                           RET
1552  0000:00FF905C  50                           PUSH EAX
1553  0000:00FF905D  53                           PUSH EBX
1554  0000:00FF905E  52                           PUSH EDX
1555  0000:00FF905F  8B 1D 9C 02 00 00            MOV EBX,SYS_FLAGS
1556  0000:00FF9065  B0 2A                        MOV AL,2AH
1557  0000:00FF9067  BA 08 F0 00 00               MOV EDX,OFFSET S_PORT_8
1558  0000:00FF906C  EE                           OUT DX,AL
1559  0000:00FF906D  B0 3A                        MOV AL,3AH
1560  0000:00FF906F  EE                           OUT DX,AL
1561  0000:00FF9070  B0 1A                        MOV AL,1AH
1562  0000:00FF9072  EE                           OUT DX,AL
1563  0000:00FF9073  B0 13                        MOV AL,13H
1564  0000:00FF9075  BA 00 F0 00 00               MOV EDX,OFFSET S_PORT_0
1565  0000:00FF907A  EE                           OUT DX,AL
1566  0000:00FF907B  C1 CB 08                     ROR EBX,08H
1567  0000:00FF907E  B0 07                        MOV AL,07H
1568  0000:00FF9080  EE                           OUT DX,AL
1569  0000:00FF9081  C1 CB 08                     ROR EBX,08H
1570  0000:00FF9084  B0 80                        MOV AL,80H
1571  0000:00FF9086  BA 10 F0 00 00               MOV EDX,OFFSET S_PORT_10
1572  0000:00FF908B  EE                           OUT DX,AL
1573  0000:00FF908C  C1 CB 08                     ROR EBX,08H
1574  0000:00FF908F  B0 BB                        MOV AL,BBH
1575  0000:00FF9091  BA 04 F0 00 00               MOV EDX,OFFSET S_PORT_4
1576  0000:00FF9096  EE                           OUT DX,AL
1577  0000:00FF9097  5A                           POP EDX
1578  0000:00FF9098  5B                           POP EBX
1579  0000:00FF9099  58                           POP EAX
1580  0000:00FF909A  C3                           RET
1581  0000:00FF909B  E8 8A 09 00 00     TAPE_WRT:       CALL 098AH
1582  0000:00FF90A0  3C 13                        CMP AL,13H
1583  0000:00FF90A2  0F 84 A5 00 00 00            JZ T_W_1
1584  0000:00FF90A8  E8 98 09 00 00               CALL 0998H
1585  0000:00FF90AD  3C 13                        CMP AL,13H
1586  0000:00FF90AF  74 EA                        JZ EAH
1587  0000:00FF90B1  E8 AA 09 00 00               CALL 09AAH
1588  0000:00FF90B6  3C 13                        CMP AL,13H
1589  0000:00FF90B8  74 EE                        JZ EEH
1590  0000:00FF90BA  8B 1D A8 02 00 00            MOV EBX,START_ADDRESS
1591  0000:00FF90C0  81 FB 00 03 00 00            CMP EBX,0300H
1592  0000:00FF90C6  0F 8C 87 00 00 00            JL T_W_2
1593  0000:00FF90CC  3B 1D 9C 00 00 00            CMP EBX,[MEM_VAL]
1594  0000:00FF90D2  0F 8F 7B 00 00 00            JG T_W_2
1595  0000:00FF90D8  8B 0D AC 02 00 00            MOV ECX,END_ADDRESS
1596  0000:00FF90DE  81 F9 00 03 00 00            CMP ECX,0300H
1597  0000:00FF90E4  0F 8C 69 00 00 00            JL T_W_2
1598  0000:00FF90EA  3B 0D 9C 00 00 00            CMP ECX,[MEM_VAL]
1599  0000:00FF90F0  0F 8F 5D 00 00 00            JG T_W_2
1600  0000:00FF90F6  2B CB                        SUB ECX,EBX
1601  0000:00FF90F8  0F 8C 5F 00 00 00            JL T_W_3
1602  0000:00FF90FE  41                           INC ECX
1603  0000:00FF90FF  8B F3                        MOV ESI,EBX
1604  0000:00FF9101  33 C0                        XOR EAX,EAX
1605  0000:00FF9103  51                           PUSH ECX
1606  0000:00FF9104  56                           PUSH ESI
1607  0000:00FF9105  02 06                        ADD AL,[ESI]
1608  0000:00FF9107  46                           INC ESI
1609  0000:00FF9108  E2 FB                        LOOP FBH
1610  0000:00FF910A  A3 B0 02 00 00               MOV 02B0H,EAX
1611  0000:00FF910F  BB EE 0E 00 00               MOV EBX,0EEEH
1612  0000:00FF9114  E8 0C 07 00 00               CALL 070CH
1613  0000:00FF9119  BE A4 02 00 00               MOV ESI,02A4H
1614  0000:00FF911E  B9 10 00 00 00               MOV ECX,10H
1615  0000:00FF9123  E8 48 00 00 00               CALL 048H
1616  0000:00FF9128  BB EE 0E 00 00               MOV EBX,0EEEH
1617  0000:00FF912D  E8 FD 06 00 00               CALL 06FDH
1618  0000:00FF9132  BB 20 00 00 00               MOV EBX,20H
```

```
1619  0000:00FF9137  E8 E9 06 00 00            CALL 06E9H
1620  0000:00FF913C  5E                        POP ESI
1621  0000:00FF913D  59                        POP ECX
1622  0000:00FF913E  E8 2D 00 00 00            CALL 2DH
1623  0000:00FF9143  BB 22 00 00 00            MOV EBX,22H
1624  0000:00FF9148  E8 E2 06 00 00            CALL 06E2H
1625  0000:00FF914D  E8 2F 08 00 00    T_W_1:  CALL 082FH
1626  0000:00FF9152  C3                        RET
1627  0000:00FF9153  BE 8A 84 FF 00    T_W_2:  MOV ESI,0FF848AH
1628  0000:00FF9158  E9 05 00 00 00            JMP 05H
1629  0000:00FF915D  BE 9B 84 FF 00    T_W_3:  MOV ESI,0FF849BH
1630  0000:00FF9162  BF EC 02 00 00            MOV EDI,02ECH
1631  0000:00FF9167  B9 11 00 00 00            MOV ECX,11H
1632  0000:00FF916C  3E F3 A4                  REP MOVSB DS:[EDI],CS:[ESI]
1633  0000:00FF916F  C3                        RET
1634  0000:00FF9170  8A 06                     MOV AL,[ESI]
1635  0000:00FF9172  51                        PUSH ECX
1636  0000:00FF9173  B4 08                     MOV AH,08H
1637  0000:00FF9175  BB 01 00 00 00    T_W_5:  MOV EBX,01H
1638  0000:00FF917A  E8 B0 06 00 00            CALL 06B0H
1639  0000:00FF917F  BB 01 00 00 00            MOV EBX,01H
1640  0000:00FF9184  E8 9C 06 00 00            CALL 069CH
1641  0000:00FF9189  BB 01 00 00 00            MOV EBX,01H
1642  0000:00FF918E  D0 D0                     RCL AL
1643  0000:00FF9190  0F 82 0A 00 00 00         JNAE T_W_4
1644  0000:00FF9196  E8 8A 06 00 00            CALL 068AH
1645  0000:00FF919B  E9 05 00 00 00            JMP 05H
1646  0000:00FF91A0  E8 8A 06 00 00    T_W_4:  CALL 068AH
1647  0000:00FF91A5  FE CC                     DEC AH
1648  0000:00FF91A7  75 CC                     JNE CCH
1649  0000:00FF91A9  59                        POP ECX
1650  0000:00FF91AA  46                        INC ESI
1651  0000:00FF91AB  E2 C3                     LOOP C3H
1652  0000:00FF91AD  C3                        RET
1653  0000:00FF91AE  E8 77 08 00 00    TAPE_RD:CALL 0877H
1654  0000:00FF91B3  3C 13                     CMP AL,13H
1655  0000:00FF91B5  0F 84 C6 00 00 00         JZ T_R_1
1656  0000:00FF91BB  B9 11 00 00 00            MOV ECX,11H
1657  0000:00FF91C0  BE 37 84 FF 00            MOV ESI,0FF8437H
1658  0000:00FF91C5  BF EC 02 00 00            MOV EDI,OFFSET DISPBF
1659  0000:00FF91CA  3E F3 A4                  REP MOVSB DS:[EDI],CS:[ESI]
1660  0000:00FF91CD  E8 95 05 00 00            CALL 0595H
1661  0000:00FF91D2  B9 EE 00 00 00            MOV ECX,EEH
1662  0000:00FF91D7  E8 F3 00 00 00            CALL F3H
1663  0000:00FF91DC  7D F4                     JNL F4H
1664  0000:00FF91DE  E2 F7                     LOOP F7H
1665  0000:00FF91E0  BF D8 02 00 00            MOV EDI,OFFSET MEM_VAL_2
1666  0000:00FF91E5  B9 10 00 00 00            MOV ECX,10H
1667  0000:00FF91EA  E8 AB 00 00 00            CALL ABH
1668  0000:00FF91EF  BE 61 84 FF 00            MOV ESI,0FF8461H
1669  0000:00FF91F4  BF EC 02 00 00            MOV EDI,OFFSET DISPBF
1670  0000:00FF91F9  B9 08 00 00 00            MOV ECX,BYTE PTR 08H
1671  0000:00FF91FE  3E F3 A4                  REP MOVSB DS:[EDI],CS:[ESI]
1672  0000:00FF9201  C6 07 7E                  MOV [EDI],BYTE PTR 7EH
1673  0000:00FF9204  B9 04 00 00 00            MOV ECX,04H
1674  0000:00FF9209  BE D8 02 00 00            MOV ESI,OFFSET MEM_VAL_2
1675  0000:00FF920E  BF FC 02 00 00            MOV EDI,02FCH
1676  0000:00FF9213  E8 D0 07 00 00            CALL 07D0H
1677  0000:00FF9218  E8 4A 05 00 00            CALL 054AH
1678  0000:00FF921D  A1 A4 02 00 00            MOV EAX,DATA_MODE
1679  0000:00FF9222  8B 1D D8 02 00 00         MOV EBX,MEM_VAL_2
1680  0000:00FF9228  3B C3                     CMP EAX,EBX
1681  0000:00FF922A  75 A6                     JNE A6H
1682  0000:00FF922C  BF A4 02 00 00            MOV EDI,OFFSET DATA_MODE
1683  0000:00FF9231  BE D8 02 00 00            MOV ESI,OFFSET MEM_VAL_2
1684  0000:00FF9236  B9 10 00 00 00            MOV ECX,10H
1685  0000:00FF923B  3E F3 A4                  REP MOVSB DS:[EDI],CS:[ESI]
1686  0000:00FF923E  A1 A8 02 00 00            MOV EAX,START_ADDRESS
1687  0000:00FF9243  8B 0D AC 02 00 00         MOV ECX,END_ADDRESS
1688  0000:00FF9249  2B C8                     SUB ECX,EAX
1689  0000:00FF924B  41                        INC ECX
1690  0000:00FF924C  8B F8                     MOV EDI,EAX
1691  0000:00FF924E  51                        PUSH ECX
1692  0000:00FF924F  57                        PUSH EDI
1693  0000:00FF9250  E8 7A 00 00 00            CALL 7AH
1694  0000:00FF9255  7D F9                     JNL F9H
1695  0000:00FF9257  E8 73 00 00 00            CALL 73H
1696  0000:00FF925C  7D F2                     JNL F2H
1697  0000:00FF925E  E8 37 00 00 00            CALL 37H
1698  0000:00FF9263  5F                        POP EDI
1699  0000:00FF9264  59                        POP ECX
1700  0000:00FF9265  33 C0                     XOR EAX,EAX
1701  0000:00FF9267  02 07                     ADD AL,[EDI]
1702  0000:00FF9269  47                        INC EDI
1703  0000:00FF926A  E2 FB                     LOOP FBH
1704  0000:00FF926C  8B 1D B0 02 00 00         MOV EBX,DEST_ADDRESS
1705  0000:00FF9272  3B C3                     CMP EAX,EBX
1706  0000:00FF9274  0F 85 0D 00 00 00         JNZ T_R_2
1707  0000:00FF927A  8B C7                     MOV EAX,EDI
1708  0000:00FF927C  A3 A0 02 00 00            MOV 02A0H,EAX
1709  0000:00FF9281  E8 FB 06 00 00    T_R_1:  CALL 06FBH
1710  0000:00FF9286  C3                        RET
1711  0000:00FF9287  B9 11 00 00 00    T_R_2:  MOV ECX,011H
1712  0000:00FF928C  BE 79 84 FF 00            MOV ESI,0FF8479H
1713  0000:00FF9291  BF EC 02 00 00            MOV EDI,02ECH
1714  0000:00FF9296  3E F3 A4                  REP MOVSB DS:[EDI],CS:[ESI]
1715  0000:00FF9299  C3                        RET
1716  0000:00FF929A  BB 00 08 00 00            MOV EBX,0800H
```

```
1717  0000:00FF929F  E8 2B 00 00 00          CALL 2BH
1718  0000:00FF92A4  7C F9                   JL F9H
1719  0000:00FF92A6  E8 24 00 00 00          CALL 24H
1720  0000:00FF92AB  0F 8C 00 00 00 00       JL T_R_3
1721  0000:00FF92B1  E8 19 00 00 00  T_R_3:  CALL 19H
1722  0000:00FF92B6  0F 8C 06 00 00 00       JL T_R_4
1723  0000:00FF92BC  F9                      STC
1724  0000:00FF92BD  E9 01 00 00 00          JMP 01H
1725  0000:00FF92C2  F8              T_R_4:  CLC
1726  0000:00FF92C3  D0 D3                   RCL BL
1727  0000:00FF92C5  FE CF                   DEC BH
1728  0000:00FF92C7  75 D6                   JNE D6H
1729  0000:00FF92C9  88 1F                   MOV [EDI],BL
1730  0000:00FF92CB  47                      INC EDI
1731  0000:00FF92CC  E2 CC                   LOOP CCH
1732  0000:00FF92CE  C3                      RET
1733  0000:00FF92CF  BA 05 E0 00 00          MOV EDX,0E005H
1734  0000:00FF92D4  EC              T_R_5:  IN AL,DX
1735  0000:00FF92D5  D0 D0                   RCL AL
1736  0000:00FF92D7  72 FB                   JNAE T_R_5
1737  0000:00FF92D9  EC                      IN AL,DX
1738  0000:00FF92DA  D0 D0                   RCL AL
1739  0000:00FF92DC  72 F6                   JNAE T_R_5
1740  0000:00FF92DE  D0 D0                   RCL AL
1741  0000:00FF92E0  EE                      OUT DX,AL
1742  0000:00FF92E1  33 C0                   XOR EAX,EAX
1743  0000:00FF92E3  FE C4           T_R_13:  INC AH
1744  0000:00FF92E5  EC                      IN AL,DX
1745  0000:00FF92E6  D0 D0                   RCL AL
1746  0000:00FF92E8  73 F9                   JNB T_R_13
1747  0000:00FF92EA  EC                      IN AL,DX
1748  0000:00FF92EB  D0 D0                   RCL AL
1749  0000:00FF92ED  73 F4                   JNB T_R_13
1750  0000:00FF92EF  D0 D0                   RCL AL
1751  0000:00FF92F1  EE                      OUT DX,AL
1752  0000:00FF92F2  80 EC 10                SUB AH,10H
1753  0000:00FF92F5  C3                      RET
1754  0000:00FF92F6  A1 A0 02 00 00  INSERT:  MOV EAX,[02A0H]
1755  0000:00FF92FB  3D 00 03 00 00          CMP EAX,0300H
1756  0000:00FF9300  0F 8C 2C 00 00 00       JL T_R_7
1757  0000:00FF9306  3B 05 9C 00 00 00       CMP EAX,[MEM_VAL]
1758  0000:00FF930C  0F 8F 20 00 00 00       JG T_R_7
1759  0000:00FF9312  8B 35 9C 00 00 00       MOV ESI,MEM_VAL
1760  0000:00FF9318  8B FE                   MOV EDI,ESI
1761  0000:00FF931A  03 3D 1C 02 00 00       ADD EDI,A_MODE
1762  0000:00FF9320  B9 01 7E 00 00          MOV ECX,07E01H
1763  0000:00FF9325  2B C8                   SUB ECX,EAX
1764  0000:00FF9327  FD                      STD
1765  0000:00FF9328  3E F3 A4                REP MOVSB DS:[EDI],CS:[ESI]
1766  0000:00FF932B  FC                      CLD
1767  0000:00FF932C  E8 50 06 00 00          CALL 0650H
1768  0000:00FF9331  C3                      RET
1769  0000:00FF9332  B9 11 00 00 00  T_R_7:  MOV ECX,011H
1770  0000:00FF9337  BE 8A 84 FF 00          MOV ESI,0FF848AH
1771  0000:00FF933C  BF EC 02 00 00          MOV EDI,02ECH
1772  0000:00FF9341  3E F3 A4                REP MOVSB DS:[EDI],CS:[ESI]
1773  0000:00FF9344  C3                      RET
1774  0000:00FF9345  8B 35 A0 02 00 00  DELETE:  MOV ESI,CURRENT_ADDRESS
1775  0000:00FF934B  81 FE 00 03 00 00       CMP ESI,0300H
1776  0000:00FF9351  0F 8C 25 00 00 00       JL T_R_6
1777  0000:00FF9357  3B 35 9C 00 00 00       CMP ESI,[MEM_VAL]
1778  0000:00FF935D  0F 8F 19 00 00 00       JG T_R_6
1779  0000:00FF9363  8B FE                   MOV EDI,ESI
1780  0000:00FF9365  03 35 1C 02 00 00       ADD ESI,A_MODE
1781  0000:00FF936B  8B 0D 9C 00 00 00       MOV ECX,MEM_VAL
1782  0000:00FF9371  2B CE                   SUB ECX,ESI
1783  0000:00FF9373  3E F3 A4                REP MOVSB DS:[EDI],CS:[ESI]
1784  0000:00FF9376  E8 06 06 00 00          CALL 0606H
1785  0000:00FF937B  C3                      RET
1786  0000:00FF937C  B9 11 00 00 00  T_R_6:  MOV ECX,011H
1787  0000:00FF9381  BE 8A 84 FF 00          MOV ESI,00FF848AH
1788  0000:00FF9386  BF EC 02 00 00          MOV EDI,OFFSET DISPBF
1789  0000:00FF938B  3E F3 A4                REP MOVSB DS:[EDI],CS:[ESI]
1790  0000:00FF938E  C3                      RET
1791  0000:00FF938F  E8 E7 06 00 00  RELTV:  CALL 06E7H
1792  0000:00FF9394  3C 13                   CMP AL,13H
1793  0000:00FF9396  0F 84 87 00 00 00       JZ T_R_12
1794  0000:00FF939C  A1 A0 02 00 00          MOV EAX,CURRENT_ADDRESS
1795  0000:00FF93A1  8B F8                   MOV EDI,EAX
1796  0000:00FF93A3  8B 0D 1C 02 00 00       MOV ECX,A_MODE
1797  0000:00FF93A9  03 C1                   ADD EAX,ECX
1798  0000:00FF93AB  8B D8                   MOV EBX,EAX
1799  0000:00FF93AD  A1 E8 02 00 00          MOV EAX,MOVE_INFO
1800  0000:00FF93B2  3D 00 03 00 00          CMP EAX,0300H
1801  0000:00FF93B7  0F 8C 6C 00 00 00       JL T_R_11
1802  0000:00FF93BD  3B 05 9C 00 00 00       CMP EAX,[MEM_VAL]
1803  0000:00FF93C3  0F 8F 60 00 00 00       JG T_R_11
1804  0000:00FF93C9  2B C3                   SUB EAX,EBX
1805  0000:00FF93CB  83 F9 04                CMP ECX,04H
1806  0000:00FF93CE  0F 84 46 00 00 00       JZ T_R_10
1807  0000:00FF93D4  83 F9 02                CMP ECX,02H
1808  0000:00FF93D7  0F 84 1D 00 00 00       JZ T_R_9
1809  0000:00FF93DD  3D 80 FF FF FF          CMP EAX,0FFFFFF80H
1810  0000:00FF93E2  0F 8C 4B 00 00 00       JL T_R_8
1811  0000:00FF93E8  3D 7F 00 00 00          CMP EAX,000000007FH
1812  0000:00FF93ED  0F 8F 40 00 00 00       JG T_R_8
1813  0000:00FF93F3  88 07                   MOV [EDI],AL
1814  0000:00FF93F5  E9 22 00 00 00          JMP 022H
```

```
0000:00FF93FA  3D 00 80 FF FF     T_R_9:    CMP EAX,0FFFF8000H
0000:00FF93FF  0F 8C 2E 00 00 00            JL T_R_8
0000:00FF9405  3D FF 7F 00 00               CMP EAX,007FFFH
0000:00FF940A  0F 8F 23 00 00 00            JG T_R_8
0000:00FF9410  88 07                        MOV [EDI],AL
0000:00FF9412  47                           INC EDI
0000:00FF9413  88 27                        MOV [EDI],AH
0000:00FF9415  E9 02 00 00 00               JMP 02H
0000:00FF941A  89 07              T_R_10:   MOV [EDI],EAX
0000:00FF941C  8B C3                        MOV EAX,EBX
0000:00FF941E  A3 A0 02 00 00               MOV CURRENT_ADDRESS,EAX
0000:00FF9423  E8 59 05 00 00     T_R_12:   CALL 0559H
0000:00FF9428  C3                           RET
0000:00FF9429  BE 8A 84 FF 00     T_R_11:   MOV ESI,0FF848AH
0000:00FF942E  E9 05 00 00 00               JMP 05H
0000:00FF9433  BE AC 84 FF 00     T_R_8:    MOV ESI,0FF84ACH
0000:00FF9438  BF EC 02 00 00               MOV EDI,02ECH
0000:00FF943D  B9 11 00 00 00               MOV ECX,011H
0000:00FF9442  3E F3 A4                     REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF9445  C3                           RET

                                  ;-------------------------------------------------------------------------
                                  ;THIS ROUTINE ALLOWS THE USER TO REVIEW AND CHANGE THE 32 BIT REGISTER FIELDS
                                  ;IN THE USER TASK STATE SEGMENT:
                                  ;-------------------------------------------------------------------------

0000:00FF9446
0000:00FF9446  33 C0                    TASK_S_32:      XOR EAX,EAX             ;CLEAR EAX
0000:00FF9448  25 FF 00 00 00           NEW_REG_32:     AND EAX,000000FFH       ;ISOLATE KEY CODE
0000:00FF944D  8B D8                                    MOV EBX,EAX             ;PUT KEY CODE IN EBX
0000:00FF944F  C1 E3 03                                 SHL EBX,03H             ;MULTIPLY BY 8
0000:00FF9452  BE CD 84 FF 00               MOV ESI,OFFSET REG_32_MSG           ;AND  GET ADDRESS OF PROPER
0000:00FF9457  03 F3                                    ADD ESI,EBX             ;DISPLAY MESSAGE
0000:00FF9459  BF EC 02 00 00               MOV EDI,OFFSET DISPBF          ;LOAD MESSAGE  IN DISPLAY BUFFER
0000:00FF945E  B9 08 00 00 00                           MOV ECX,00000008H
0000:00FF9463  3E F3 A4                                 REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF9466  C6 07 7F                     MOV [EDI],BYTE PTR 7FH          ;LOAD ARROW IN DISPLAY BUFFER
0000:00FF9469
0000:00FF9469  8B D8                                    MOV EBX,EAX             ;PUT KEY CODE IN EBX
0000:00FF946B  C1 E3 02                                 SHL EBX,02H             ;MULTIPLY BY 4
0000:00FF946E  BE 3C 02 00 00               MOV ESI,OFFSET USER_CR3             ;AND GET ADDRESS OF FIELD IN
0000:00FF9473  03 F3                                    ADD ESI,EBX             ;USER TSS
0000:00FF9475  8B EE                                    MOV EBP,ESI
0000:00FF9477  BF FC 02 00 00                           MOV EDI,OFFSET DISPBF+16
0000:00FF947C  B9 04 00 00 00                           MOV ECX,00000004H
0000:00FF9481  E8 62 05 00 00             CALL FORMAT                       ;FORMAT REGISTER FIELD FOR DISPLAY
0000:00FF9486  E8 E9 03 00 00     GET_REG_32:     CALL KEYBOARD             ;GET KEY PRESS
0000:00FF948B  3C 19                        CMP AL,19H                      ;IF INS PRESSED, GO TO CHANGE_REG_2
0000:00FF948D  0F 84 0E 00 00 00                        JZ CHANGE_REG_32
0000:00FF9493  3C 17                        CMP AL,17H                  ;IF + PRESSED, GO TO END_TASK_STATE_
0000:00FF9495  0F 84 4D 00 00 00                        JZ END_TASK_STATE_32
0000:00FF949B  3C 0A                        CMP AL,0AH                  ;IF KEY CODE > 0A, GETNEW KEY PRESS
0000:00FF949D  7F E7                                    JG GET_REG_32
0000:00FF949F  EB A7                                    JMP NEW_REG_32              ;DISPLAY NEW REGISTER
0000:00FF94A1
0000:00FF94A1  BF F4 02 00 00     CHANGE_REG_32:  MOV EDI,OFFSET DISPBF+8     ;PUT ARROW  IN DISPLAY BUFFER
0000:00FF94A6  C6 07 7E                           MOV [EDI],BYTE PTR 7EH
0000:00FF94A9  8B F5              NEXT_32_DIGIT:  MOV ESI,EBP                 ;FORMAT REGISTER FIELD FOR DISPLAY
0000:00FF94AB  BF FC 02 00 00                     MOV EDI,OFFSET DISPBF+16
0000:00FF94B0  B9 04 00 00 00                     MOV ECX,00000004H
0000:00FF94B5  E8 2E 05 00 00                     CALL  FORMAT
0000:00FF94BA  E8 B5 03 00 00     NEXT_32_INPUT:  CALL KEYBOARD                   ;GET  NEXT KEY
0000:00FF94BF  3C 17                              CMP AL,17H                    ;IF + KEY PRESSED, GO BACK T
0000:00FF94C1  0F 84 17 00 00 00                  JZ REG_MODE_32
0000:00FF94C7  3C 0F                              CMP AL,0FH                  ;IF KEY CODE >  0FH, GET NEW KEY
0000:00FF94C9  7F EF                              JG NEXT_32_INPUT
0000:00FF94CB  8B F5                              MOV ESI,EBP
0000:00FF94CD  8B 1E                              MOV EBX,[ESI]                       ;ELSE GET  REG DATA
0000:00FF94CF  C1 C3 04                           ROL EBX,04H                         ;ROLL LEFT 4 BITS
0000:00FF94D2  81 E3 F0 FF FF FF                  AND EBX,FFFFFFF0H                   ;DELETE LOWER 4 BITS
0000:00FF94D8  0A D8                              OR BL,AL                            ;INSERT NEW DATA
0000:00FF94DA  89 1E                              MOV [ESI],EBX               ;LOAD INTO REGISTER FIELD OF TSS
0000:00FF94DC  EB CB                              JMP NEXT_32_DIGIT                   ;GET NEXT DIGIT
0000:00FF94DE
0000:00FF94DE  BE F4 02 00 00     REG_MODE_32:    MOV ESI,OFFSET DISPBF+8       ;LOAD ARROW  INTO DISPLAY BUFF
0000:00FF94E3  C6 06 7F                           MOV [ESI],BYTE PTR 7FH
0000:00FF94E6  EB 9E                              JMP GET_REG_32                      ;GO TO GET_REG_32
0000:00FF94E8
0000:00FF94E8                     END_TASK_STATE_32:
0000:00FF94E8  E8 94 04 00 00                     CALL DISPLAY_INFO           ;DISPLAY CURRENT ADDRERSS AND DATA
0000:00FF94ED  C3                                 RET
                                  ;-------------------------------------------------------------------------
                                  ;THIS ROUTINE ALLOWS THE USER TO REVIEW AND CHANGE THE DATA IN THE 16 BIT
                                  ;REGISTER FIELDS OF THE USER TASK STATE SEGMENT
                                  ;-------------------------------------------------------------------------

0000:00FF94EE  33 C0              TASK_S_16:      XOR EAX,EAX                     ;CLEAR EAX
0000:00FF94F0  25 FF 00 00 00     NEW_REG_16:     AND EAX,000000FFH               ;ISOLATE KEY CODE
0000:00FF94F5  8B D8                              MOV EBX,EAX                     ;PUT KEY CODE IN EBX
0000:00FF94F7  C1 E3 03                           SHL EBX,03H                     ;MULTIPLY BY 8
0000:00FF94FA  BE 25 85 FF 00                     MOV ESI,OFFSET REG_16_MSG       ;GET ADDRESS OF PROPER
0000:00FF94FF  03 F3                              ADD ESI,EBX                     ;DISPLAY MESSAGE
0000:00FF9501  BF EC 02 00 00                     MOV EDI,OFFSET DISPBF           ;LOAD IT INTO DISPLAY BUFFER
0000:00FF9506  B9 08 00 00 00                     MOV ECX,00000008H
0000:00FF950B  3E F3 A4                           REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF950E  C6 07 7F                           MOV [EDI],BYTE PTR 7FH          ;LD ARROW INTO DISP BUFFER
0000:00FF9511  47                                 INC EDI
0000:00FF9512  BE 59 84 FF 00                     MOV ESI,OFFSET X_MSG            ;LD xxxxxxxx INTO DISP BUFF
```

```
1913  0000:00FF9517  B9 08 00 00 00                  MOV ECX,00000008H
1914  0000:00FF951C  3E F3 A4                        REP MOVSB DS:[EDI],CS:[ESI]
1915  0000:00FF951F
1916  0000:00FF951F  8B D8                           MOV EBX,EAX                ;PUT KEY CODE INTO EBX
1917  0000:00FF9521  C1 E3 02                        SHL EBX,02H                ;MULTIPLY BY 4
1918  0000:00FF9524  BE 68 02 00 00                  MOV ESI,OFFSET USER_ES     ;GET ADDRESS OF REGISTER FIELD OF
1919  0000:00FF9529  03 F3                           ADD ESI,EBX                ;USER TSS
1920  0000:00FF952B  8B EE                           MOV EBP,ESI
1921  0000:00FF952D  BF FC 02 00 00                  MOV EDI,OFFSET DISPBF+16
1922  0000:00FF9532  B9 02 00 00 00                  MOV ECX,00000002H
1923  0000:00FF9537  E8 AC 04 00 00                  CALL FORMAT                      ;FORMAT REG DATA FOR DISPLAY
1924  0000:00FF953C  E8 33 03 00 00     GET_REG_16:  CALL KEYBOARD                    ;GET NEXT KEY PRESS
1925  0000:00FF9541  3C 19                           CMP AL,19H                     ;IF INS  WAS PRESSED
1926  0000:00FF9543  0F 84 0E 00 00 00               JZ CHANGE_REG_16               ;GO TO CHANGE_REG_16
1927  0000:00FF9549  3C 17                           CMP AL,17H                     ;IF TS16 WAS PRESSED
1928  0000:00FF954B  0F 84 55 00 00 00               JZ END_TASK_STATE_16           ;GO TO END_TASK_STATE_16
1929  0000:00FF9551  3C 06                           CMP AL,06H              ;IF KEY CODE > 06H, GET NEW KEY CODE
1930  0000:00FF9553  7F E7                           JG GET_REG_16
1931  0000:00FF9555  EB 99                           JMP NEW_REG_16                 ;ELSE DISPLAY REG DATA
1932  0000:00FF9557
1933  0000:00FF9557  BF F4 02 00 00      CHANGE_REG_16:  MOV EDI,OFFSET DISPBF+8     ;LOAD ARROW INTO DISPLAY
1934  0000:00FF955C  C6 07 7E                        MOV [EDI],BYTE PTR 7EH
1935  0000:00FF955F  8B F5                           NEXT_16_DIGIT:  MOV ESI,EBP    ;FMT REG DATA FOR DISP
1936  0000:00FF9561  BF FC 02 00 00                  MOV EDI,OFFSET DISPBF+16
1937  0000:00FF9566  B9 02 00 00 00                  MOV ECX,00000002H
1938  0000:00FF956B  E8 78 04 00 00                  CALL  FORMAT
1939  0000:00FF9570  E8 FF 02 00 00     NEXT_16_INPUT:  CALL KEYBOARD               ;GET NEXT KEY PRESS
1940  0000:00FF9575  3C 17                           CMP AL,17H                     ;IF +, GO  TO REG_MODE_16
1941  0000:00FF9577  0F 84 1F 00 00 00               JZ REG_MODE_16
1942  0000:00FF957D  3C 0F                           CMP AL,0FH                     ;IF GREATER THAN 0F
1943  0000:00FF957F  7F EF                           JG NEXT_16_INPUT               ;GET NEW KEY CODE
1944  0000:00FF9581  8B F5                           MOV ESI,EBP                    ;GET REG FIELD DATA
1945  0000:00FF9583  66                              DB DATA_SZ
1946  0000:00FF9584  8A 1E                           MOV BL,[ESI]
1947  0000:00FF9586  46                              INC ESI
1948  0000:00FF9587  8A 3E                           MOV BH,[ESI]
1949  0000:00FF9589  C1 C3 04                        ROL EBX,04H                        ;ROLL LEFT 4 BITS
1950  0000:00FF958C  81 E3 F0 FF FF FF               AND EBX,FFFFFFF0H                  ;CLEAR LOWER 4 BITS
1951  0000:00FF9592  0A D8                           OR BL,AL                           ;INSET NEW DATA
1952  0000:00FF9594  66                              DB DATA_SZ                         ;LOAD  INTO REG FIELD
1953  0000:00FF9595  88 3E                           MOV [ESI],BH
1954  0000:00FF9597  4E                              DEC ESI
1955  0000:00FF9598  88 1E                           MOV [ESI],BL
1956  0000:00FF959A  EB C3                           JMP NEXT_16_DIGIT                  ;GET NEXT DIGIT
1957  0000:00FF959C
1958  0000:00FF959C  BE F4 02 00 00     REG_MODE_16:  MOV ESI,OFFSET DISPBF+8       ;LOAD ARROW INTO DISPLAY
1959  0000:00FF95A1  C6 06 7F                        MOV [ESI],BYTE PTR 7FH
1960  0000:00FF95A4  EB 96                           JMP GET_REG_16                     ;GO TO GET_REG_16
1961  0000:00FF95A6
1962  0000:00FF95A6                    END_TASK_STATE_16:
1963  0000:00FF95A6  E8 D6 03 00 00                  CALL DISPLAY_INFO          ;DISPLAY CURRENT ADDRESS AND DATA
1964  0000:00FF95AB  C3                              RET
1965                                   ;-------------------------------------------------------------------------
1966  0000:00FF95AC  0F 21 F8          BREAK_POINT:  MOV EAX,DR7
1967  0000:00FF95AF  25 AA 02 00 00                  AND EAX,000002AAH
1968  0000:00FF95B4  0D 00 02 00 00                  OR EAX,00000200H
1969  0000:00FF95B9  0F 23 F8                        MOV DR7,EAX
1970  0000:00FF95BC  BE C5 84 FF 00                  MOV ESI,OFFSET BRPT_MSG
1971  0000:00FF95C1  BF EC 02 00 00                  MOV EDI,OFFSET DISPBF
1972  0000:00FF95C6  B9 08 00 00 00                  MOV ECX,08H
1973  0000:00FF95CB  3E F3 A4                        REP MOVSB DS:[EDI],CS:[ESI]
1974  0000:00FF95CE  33 D2                           XOR EDX,EDX
1975  0000:00FF95D0
1976  0000:00FF95D0  BB 1F 84 FF 00     NEW_BP:       MOV EBX,OFFSET HEXCODES
1977  0000:00FF95D5  BF F1 02 00 00                  MOV EDI,OFFSET DISPBF+5
1978  0000:00FF95DA  8A C2                           MOV AL,DL
1979  0000:00FF95DC  D7                              XLAT
1980  0000:00FF95DD  88 07                           MOV [EDI],AL
1981  0000:00FF95DF  BF F4 02 00 00                  MOV EDI,OFFSET DISPBF+8
1982  0000:00FF95E4  C6 07 7E                        MOV [EDI],BYTE PTR 7EH
1983  0000:00FF95E7  0F 21 F8                        MOV EAX,DR7
1984  0000:00FF95EA  B3 02                           MOV BL,02H
1985  0000:00FF95EC  8B CA                           MOV ECX,EDX
1986  0000:00FF95EE  80 F9 00                        CMP CL,00H
1987  0000:00FF95F1  0F 84 05 00 00 00               JZ TEST_SET
1988  0000:00FF95F7  C0 E3 02          GET_MASK:     SHL BL,02H
1989  0000:00FF95FA  E2 FB                           LOOP GET_MASK
1990  0000:00FF95FC  BF F3 02 00 00     TEST_SET:     MOV EDI,OFFSET DISPBF+7
1991  0000:00FF9601  22 C3                           AND AL,BL
1992  0000:00FF9603  3C 00                           CMP AL,00H
1993  0000:00FF9605  0F 84 08 00 00 00               JZ  NOT_SET
1994  0000:00FF960B  C6 07 53                        MOV [EDI],BYTE PTR 53H
1995  0000:00FF960E  E9 03 00 00 00                  JMP GET_BP_DATA
1996  0000:00FF9613  C6 07 43          NOT_SET:      MOV [EDI],BYTE PTR 43H
1997  0000:00FF9616  BF CE 96 FF 00     GET_BP_DATA:  MOV EDI,OFFSET GET_DRX
1998  0000:00FF961B  8B CA                           MOV ECX,EDX
1999  0000:00FF961D  C1 E1 02                        SHL ECX,02H
2000  0000:00FF9620  03 F9                           ADD EDI,ECX
2001  0000:00FF9622  FF D7                           CALL EDI
2002  0000:00FF9624  A3 B0 02 00 00                  MOV TEMP_0,EAX
2003  0000:00FF9629  BE B0 02 00 00                  MOV ESI,OFFSET TEMP_0
2004  0000:00FF962E  BF FC 02 00 00                  MOV EDI,OFFSET DISPBF+16
2005  0000:00FF9633  B9 04 00 00 00                  MOV ECX,00000004H
2006  0000:00FF9638  E8 AB 03 00 00                  CALL FORMAT
2007  0000:00FF963D  E8 32 02 00 00     NEW_BP_KEY:   CALL KEYBOARD
2008  0000:00FF9642  3C 0F                           CMP AL,0FH
2009  0000:00FF9644  0F 8E 5D 00 00 00               JLE CHANGE_BP
2010  0000:00FF964A  3C 17                           CMP AL,17H
```



```
2011  0000:00FF964C  0F 84 35 00 00 00              JZ UP_BP
2012  0000:00FF9652  3C 13                         CMP AL,13H
2013  0000:00FF9654  0F 84 3D 00 00 00              JZ DOWN_BP
2014  0000:00FF965A  3C 12                         CMP AL,12H
2015  0000:00FF965C  0F 84 14 00 00 00              JZ CLEAR_BP
2016  0000:00FF9662  3C 11                         CMP AL,11H
2017  0000:00FF9664  0F 84 02 00 00 00              JZ SET_BP
2018  0000:00FF966A  EB D1                         JMP NEW_BP_KEY
2019  0000:00FF966C
2020  0000:00FF966C  0F 21 F8          SET_BP:     MOV EAX,DR7
2021  0000:00FF966F  0A C3                         OR AL,BL
2022  0000:00FF9671  0F 23 F8                      MOV DR7,EAX
2023  0000:00FF9674  EB 86                         JMP TEST_SET
2024  0000:00FF9676
2025  0000:00FF9676  0F 21 F8          CLEAR_BP:   MOV EAX,DR7
2026  0000:00FF9679  F6 D3                         NOT BL
2027  0000:00FF967B  22 C3                         AND AL,BL
2028  0000:00FF967D  F6 D3                         NOT BL
2029  0000:00FF967F  0F 23 F8                      MOV DR7,EAX
2030  0000:00FF9682  E9 75 FF FF FF                JMP TEST_SET
2031  0000:00FF9687
2032  0000:00FF9687  FE C2              UP_BP:      INC DL
2033  0000:00FF9689  80 FA 03                      CMP DL,03H
2034  0000:00FF968C  0F 8F 5C 00 00 00             JG LEAVE_BP
2035  0000:00FF9692  E9 39 FF FF FF                JMP NEW_BP
2036  0000:00FF9697
2037  0000:00FF9697  FE CA              DOWN_BP:    DEC DL
2038  0000:00FF9699  80 FA 00                      CMP DL,00H
2039  0000:00FF969C  0F 8C 4C 00 00 00             JL LEAVE_BP
2040  0000:00FF96A2  E9 29 FF FF FF                JMP NEW_BP
2041  0000:00FF96A7
2042  0000:00FF96A7  BE B0 02 00 00     CHANGE_BP:  MOV ESI,OFFSET TEMP_0
2043  0000:00FF96AC  8B 0E                         MOV ECX,[ESI]
2044  0000:00FF96AE  C1 C1 04                      ROL ECX,04H
2045  0000:00FF96B1  81 E1 F0 FF FF FF             AND  ECX,FFFFFFF0H
2046  0000:00FF96B7  0A C8                         OR CL,AL
2047  0000:00FF96B9  8B C1                         MOV EAX,ECX
2048  0000:00FF96BB  BF DE 96 FF 00                MOV EDI,OFFSET PUT_DRX
2049  0000:00FF96C0  8B CA                         MOV ECX,EDX
2050  0000:00FF96C2  C1 E1 02                      SHL ECX,02H
2051  0000:00FF96C5  03 F9                         ADD EDI,ECX
2052  0000:00FF96C7  FF D7                         CALL EDI
2053  0000:00FF96C9  E9 48 FF FF FF                JMP GET_BP_DATA
2054  0000:00FF96CE
2055  0000:00FF96CE
2056  0000:00FF96CE  0F 21 C0           GET_DRX:    MOV EAX,DR0
2057  0000:00FF96D1  C3                            RET
2058  0000:00FF96D2  0F 21 C8                      MOV EAX,DR1
2059  0000:00FF96D5  C3                            RET
2060  0000:00FF96D6  0F 21 D0                      MOV EAX,DR2
2061  0000:00FF96D9  C3                            RET
2062  0000:00FF96DA  0F 21 D8                      MOV EAX,DR3
2063  0000:00FF96DD  C3                            RET
2064  0000:00FF96DE
2065  0000:00FF96DE  0F 23 C0           PUT_DRX:    MOV DR0,EAX
2066  0000:00FF96E1  C3                            RET
2067  0000:00FF96E2  0F 23 C8                      MOV DR1,EAX
2068  0000:00FF96E5  C3                            RET
2069  0000:00FF96E6  0F 23 D0                      MOV DR2,EAX
2070  0000:00FF96E9  C3                            RET
2071  0000:00FF96EA  0F 23 D8                      MOV DR3,EAX
2072  0000:00FF96ED  C3                            RET
2073  0000:00FF96EE
2074  0000:00FF96EE  0F 21 C0           LEAVE_BP:   MOV EAX,DR0
2075  0000:00FF96F1  A3 B4 02 00 00                MOV DEBUG_REG_0,EAX
2076  0000:00FF96F6  0F 21 C8                      MOV EAX,DR1
2077  0000:00FF96F9  A3 B8 02 00 00                MOV DEBUG_REG_1,EAX
2078  0000:00FF96FE  0F 21 D0                      MOV EAX,DR2
2079  0000:00FF9701  A3 BC 02 00 00                MOV DEBUG_REG_2,EAX
2080  0000:00FF9706  0F 21 D8                      MOV EAX,DR3
2081  0000:00FF9709  A3 C0 02 00 00                MOV DEBUG_REG_3,EAX
2082  0000:00FF970E  0F 21 F8                      MOV EAX,DR7
2083  0000:00FF9711  A3 C4 02 00 00                MOV DEBUG_REG_7,EAX
2084  0000:00FF9716
2085  0000:00FF9716  E8 66 02 00 00                CALL DISPLAY_INFO
2086  0000:00FF971B  C3                            RET
2087  0000:00FF971C
2088                                 ;-------------------------------------------------------------------
2089                                        ;SCAN: KEYBOARD ROUTINE - THIS ROUTINE SCANS THE KEYBOARD ONCE
2090                                 ;-------------------------------------------------------------------
2091  0000:00FF971C  53                     SCAN:       PUSH EBX
2092  0000:00FF971D  51                                 PUSH ECX
2093  0000:00FF971E  52                                 PUSH EDX
2094  0000:00FF971F  55                                 PUSH EBP
2095  0000:00FF9720
2096  0000:00FF9720  BD 00 00 00 00                     MOV EBP,0H          ;USE EBP TO STORE KEY STAT
2097  0000:00FF9725  33 C0                              XOR EAX,EAX         ;CLEAR  EAX
2098                                                                        ;INITIALIZE COLUMN STROBER (BH)
2099  0000:00FF9727  BB 08 01 00 00                     MOV EBX,108H        ;AND ROW  COUNTER (BL)
2100  0000:00FF972C  B9 A4 01 00 00             COLS:   MOV ECX,01A4H       ;SET DELAY COUNTER
2101  0000:00FF9731  E2 FE                       DELAY:  LOOP DELAY          ;LOOP  FOR DEBOUNCE
2102  0000:00FF9733  F6 D7                              NOT BH              ;SET SELECTED COLUMN LOW,
2103                                                                        ;ALL OTHERS HIGH
2104  0000:00FF9735  8A C7                              MOV AL,BH           ;OUTPUT BH TO ACTIVATE COLUMN
2105  0000:00FF9737  BA 04 E0 00 00                     MOV EDX,OFFSET PORTC_L
2106  0000:00FF973C  EE                                 OUT DX,AL
2107  0000:00FF973D
2108  0000:00FF973D  B9 05 00 00 00                     MOV ECX,05H         ;FIVE KEYS IN COLUMN TO BE READ
```



```
2109  0000:00FF9742  BA 00 E0 00 00                        MOV EDX,OFFSET PORTA_L
2110  0000:00FF9747  EC                                    IN AL,DX              ;READ IN KEY STATUS FOR COLUMN
2111  0000:00FF9748  FE C4                     ROWS:       INC AH
2112  0000:00FF974A  D0 D8                                 RCR AL
2113  0000:00FF974C  0F 82 02 00 00 00                     JNAE SKIP                   ;CHECK STATUS OF EACH KEY
2114  0000:00FF9752  8B E8                                 MOV EBP,EAX
2115  0000:00FF9754  E2 F2                     SKIP:       LOOP ROWS
2116  0000:00FF9756  F6 D7                                 NOT BH
2117  0000:00FF9758  D0 C7                                 ROL BH                      ;UPDATE DIGIT STROBER
2118  0000:00FF975A  FE CB                                 DEC BL                      ;DECREMENT DIGIT COUNTER
2119  0000:00FF975C  75 CE                                 JNZ COLS                    ;READ NEXT COLUMN
2120  0000:00FF975E  8B C5                                 MOV EAX,EBP                 ;PUT RESULTS IN AX
2121  0000:00FF9760  86 C4                                 XCHG AL,AH
2122  0000:00FF9762
2123  0000:00FF9762  5D                                    POP EBP
2124  0000:00FF9763  5A                                    POP EDX
2125  0000:00FF9764  59                                    POP ECX
2126  0000:00FF9765  5B                                    POP EBX
2127  0000:00FF9766
2128  0000:00FF9766  C3                     GET_BACK:      RET
2129  0000:00FF9767
2130  0000:00FF9767
2131                                 ;-------------------------------------------------------------------------
2132                                 ;THIS ROUTINE WRITES THE CONTENTS OF THE DISPLAY BUFFER TO THE DISPLAY
2133                                 ;-------------------------------------------------------------------------
2134  0000:00FF9767
2135  0000:00FF9767  50                     LCD_DISPLAY:   PUSH EAX
2136  0000:00FF9768  51                                    PUSH ECX
2137  0000:00FF9769  52                                    PUSH EDX
2138  0000:00FF976A  56                                    PUSH ESI
2139  0000:00FF976B
2140  0000:00FF976B  E8 85 00 00 00                        CALL BUSY             ;WAIT TILL DISPLAY NOT BUSY
2141  0000:00FF9770  B0 30                                 MOV AL,30H            ;WRITE FUNCTION SET COMMAND
2142                                                                             ;8 BITS DATA, 2LINES, 5x7 FONT
2143  0000:00FF9772  BA 02 E0 00 00                        MOV EDX,OFFSET PORTB_L
2144  0000:00FF9777  EE                                    OUT DX,AL
2145  0000:00FF9778  B0 F1                                 MOV AL,F1H                  ;RAISE E LINE
2146  0000:00FF977A  BA 05 E0 00 00                        MOV EDX,OFFSET PORTC_H
2147  0000:00FF977F  EE                                    OUT DX,AL
2148  0000:00FF9780  B0 F3                                 MOV AL,F3H                  ;LOWER E LINE
2149  0000:00FF9782  EE                                    OUT DX,AL
2150  0000:00FF9783
2151  0000:00FF9783  E8 6D 00 00 00                        CALL BUSY              ;WAIT TILL DISPLAY NOT BUSY
2152  0000:00FF9788  B0 0C                                 MOV AL,0CH             ;WRITE DISPLAY ON/OFF COMMAND
2153                                                                              ;TURN ON, NO CURSOR, NO BLINK
2154  0000:00FF978A  BA 02 E0 00 00                        MOV EDX,OFFSET PORTB_L
2155  0000:00FF978F  EE                                    OUT DX,AL
2156  0000:00FF9790  B0 F1                                 MOV AL,F1H                  ;RAISE E LINE
2157  0000:00FF9792  BA 05 E0 00 00                        MOV EDX,OFFSET PORTC_H
2158  0000:00FF9797  EE                                    OUT DX,AL
2159  0000:00FF9798  B0 F3                                 MOV AL,F3H                  ;LOWER E LINE
2160  0000:00FF979A  EE                                    OUT DX,AL
2161  0000:00FF979B
2162  0000:00FF979B  E8 55 00 00 00                        CALL BUSY              ;WAIT TILL DISPLAY NOT BUSY
2163  0000:00FF97A0  B0 06                                 MOV AL,06H             ;WRITE ENTRY MODE SET COMMAND
2164                                                                              ;INC DD RAM ADDRESS, NO SHIFT
2165  0000:00FF97A2  BA 02 E0 00 00                        MOV EDX,OFFSET PORTB_L
2166  0000:00FF97A7  EE                                    OUT DX,AL
2167  0000:00FF97A8  B0 F1                                 MOV AL,F1H                  ;RAISE E LINE
2168  0000:00FF97AA  BA 05 E0 00 00                        MOV EDX,OFFSET PORTC_H
2169  0000:00FF97AF  EE                                    OUT DX,AL
2170  0000:00FF97B0  B0 F3                                 MOV AL,F3H                  ;LOWER E LINE
2171  0000:00FF97B2  EE                                    OUT DX,AL
2172  0000:00FF97B3
2173  0000:00FF97B3  E8 3D 00 00 00                CALL BUSY                   ;WAIT TILL DISPLAY NOT BUSY
2174  0000:00FF97B8  B0 8C                         MOV AL,8CH                  ;WRITE SET DD RAM ADDRESS COMMAND
2175                                                                           ;FIRST LINE, 15TH CHARACTER
2176  0000:00FF97BA  BA 02 E0 00 00                        MOV EDX,OFFSET PORTB_L
2177  0000:00FF97BF  EE                                    OUT DX,AL
2178  0000:00FF97C0  B0 F1                                 MOV AL,F1H                  ;RAISE E LINE
2179  0000:00FF97C2  BA 05 E0 00 00                        MOV EDX,OFFSET PORTC_H
2180  0000:00FF97C7  EE                                    OUT DX,AL
2181  0000:00FF97C8  B0 F3                                 MOV AL,F3H                  ;LOWER E LINE
2182  0000:00FF97CA  EE                                    OUT DX,AL
2183  0000:00FF97CB
2184  0000:00FF97CB  B9 11 00 00 00                    MOV ECX,17                      ;SET FOR EIGHT CHARACTERS
2185  0000:00FF97D0  BE EC 02 00 00                    MOV ESI,OFFSET DISPBF           ;PUT END DISPLAY BUFFER
2186                                                                                   ;ADDRESS IN SI
2187  0000:00FF97D5  E8 1B 00 00 00  WRITE_CHAR:       CALL BUSY
2188  0000:00FF97DA  8A 06                             MOV AL,[ESI]                    ;GET CHARACTER
2189  0000:00FF97DC  BA 02 E0 00 00                    MOV EDX,OFFSET PORTB_L
2190  0000:00FF97E1  EE                                OUT DX,AL                       ;WRITE CHARACTER
2191  0000:00FF97E2  B0 F9                             MOV AL,F9H                      ;RAISE E LINE
2192  0000:00FF97E4  BA 05 E0 00 00                    MOV EDX,OFFSET PORTC_H
2193  0000:00FF97E9  EE                                OUT DX,AL
2194  0000:00FF97EA  B0 FB                             MOV AL,FBH                      ;LOWER E LINE
2195  0000:00FF97EC  EE                                OUT DX,AL
2196  0000:00FF97ED  46                                INC ESI
2197  0000:00FF97EE
2198  0000:00FF97EE  E2 E5                             LOOP WRITE_CHAR
2199  0000:00FF97F0
2200  0000:00FF97F0  5E                                POP ESI
2201  0000:00FF97F1  5A                                POP EDX
2202  0000:00FF97F2  59                                POP ECX
2203  0000:00FF97F3  58                                POP EAX
2204  0000:00FF97F4
2205  0000:00FF97F4  C3                                RET
2206  0000:00FF97F5
```

```
0000:00FF97F5  B0 92              BUSY:          MOV AL,92H                    ;SET LOWER PORT B FOR INPUT
0000:00FF97F7  BA 06 E0 00 00                    MOV EDX,OFFSET CONTR_L
0000:00FF97FC  EE                                OUT DX,AL
0000:00FF97FD  B0 F5              STILL_BUSY:    MOV AL,F5H                        ;RAISE E, READ DATA
0000:00FF97FF  C7 C2 05 E0 00 00                 MOV EDX,PORTC_H
0000:00FF9805  EE                                OUT DX,AL
0000:00FF9806
0000:00FF9806  C7 C2 02 E0 00 00                 MOV EDX,PORTB_L                   ;INPUT
0000:00FF980C  EC                                IN AL,DX                          ;INPUT DATA
0000:00FF980D  8A E0                             MOV AH,AL
0000:00FF980F
0000:00FF980F  B0 F7                             MOV AL,F7H                        ;LOWER E LINE
0000:00FF9811  BA 05 E0 00 00                    MOV EDX,OFFSET PORTC_H
0000:00FF9816  EE                                OUT DX,AL
0000:00FF9817
0000:00FF9817  80 E4 80                          AND AH,80H
0000:00FF981A  75 E1                             JNZ STILL_BUSY
0000:00FF981C
0000:00FF981C  B0 90                             MOV AL,90H                       ;SET PORT B FOR OUTPUT
0000:00FF981E  BA 06 E0 00 00                    MOV EDX,OFFSET CONTR_L
0000:00FF9823  EE                                OUT DX,AL
0000:00FF9824
0000:00FF9824  C3                                RET
                                ;-------------------------------------------------------------------------
                                 ;TONE GENERATION ROUTINES
                                ;-------------------------------------------------------------------------
                                 ;
                                 ;          BEEP     -   GENERATES TONE WITH THE FREQUENCY AND DURATION
                                 ;                        CURRENTLY LOADED INTO MEMORY LOCATIONS FREQ AND
                                 ;                        DURATION
                                 ;
                                 ;          TONE     -   GENERATES TONE WITH THE FREQUENCY SPECIFIED BY ECX
                                 ;                        AND DURATION SPECIFIED BY EDX
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9825  B9 22 00 00 00            MOV ECX,0022H
0000:00FF982A  E9 2C 00 00 00            JMP 02CH
0000:00FF982F  B9 44 00 00 00            MOV ECX,0044H
0000:00FF9834  E9 22 00 00 00            JMP 022H
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9839
0000:00FF9839  53                 BEEP:          PUSH EBX
0000:00FF983A  51                                PUSH ECX
0000:00FF983B  8B 0D 18 02 00 00                 MOV ECX,FREQ
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9841  81 E1 FF FF 00 00         AND ECX,0000FFFFH
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9847  8B 1D 1A 02 00 00                 MOV EBX,DURATION
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF984D  81 E3 FF FF 00 00         AND EBX,0000FFFFH
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9853  E8 03 00 00 00                    CALL TONE
0000:00FF9858  59                                POP ECX
0000:00FF9859  5B                                POP EBX
0000:00FF985A  C3                                RET
0000:00FF985B
0000:00FF985B  50                 TONE:          PUSH EAX
0000:00FF985C  52                                PUSH EDX
0000:00FF985D
0000:00FF985D  03 DB                             ADD EBX,EBX                  ;DOUBLE # OF PERIODS TO
                                                                              ;GET # OF 1/2 PERIODS
0000:00FF985F  B0 AA                             MOV AL,AAH                   ;LOAD AL 10101010
0000:00FF9861  C7 C2 05 E0 00 00                 MOV EDX,PORTC_H
0000:00FF9867
0000:00FF9867  EE                 NEXT_PERIOD:   OUT DX,AL                    ;OUTPUT BIT TO SPEAKER
0000:00FF9868  D0 C0                             ROL AL                       ;GET NEXT BIT TO BE OUTPUT
0000:00FF986A
0000:00FF986A  51                                PUSH ECX
0000:00FF986B  E2 FE                    FREQ_DELAY:  LOOP FREQ_DELAY          ;TIME BETWEEN 1/2 PERIODS IS
0000:00FF986D  59                                POP ECX                      ;DETERMINED BY FREQ
0000:00FF986E
0000:00FF986E  4B                                DEC EBX                     ;1/2 PERIOD HAS BEEN GENERATED
0000:00FF986F  75 F6                             JNZ NEXT_PERIOD
0000:00FF9871
0000:00FF9871  5A                                POP EDX                      ;OTHERWISE, TONE IS COMPLETED
0000:00FF9872  58                                POP EAX
0000:00FF9873  C3                                RET
0000:00FF9874
                                ;-------------------------------------------------------------------------
                                        ;KEYBOARD - UPDATES THE DISPLAY AND SCANS  THE KEYBOARD UNTIL A KEY IS
                                        ;PRESSED. THE KEYCODE OF THE KEY PRESSED  IS RETURNED  IN AL
                                ;-------------------------------------------------------------------------
0000:00FF9874
0000:00FF9874  53                       KEYBOARD:      PUSH EBX
0000:00FF9875  51                                      PUSH ECX
0000:00FF9876  55                                      PUSH EBP
0000:00FF9877  56                                      PUSH ESI
0000:00FF9878  57                                      PUSH EDI
0000:00FF9879
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF9879  32 C0                     XOR AL,AL
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++
0000:00FF987B  BD 2F 99 FF 00                          MOV EBP,OFFSET KEYTABLE
0000:00FF9880  E8 E2 FE FF FF           UPDATE_DISPLAY: CALL LCD_DISPLAY          ;UPDATE DISPLAY
0000:00FF9885  E8 92 FE FF FF           GET_INPUT:      CALL SCAN                 ;SCAN KEYBOARD
0000:00FF988A  3C 00                                   CMP AL,00H                ;WAS KEY PRESSED?
0000:00FF988C  0F 85 0A 00 00 00                       JNZ COMPAIR_LAST
```



```
2305  0000:00FF9892  33 C0                              XOR EAX,EAX                    ;IF NOT
2306  0000:00FF9894  67                                 DB ADDR_SZ                     ;CLEAR  LAST_KEY
2307  0000:00FF9895  A3 98 02 00 00                     MOV LAST_KEY,EAX
2308  0000:00FF989A  EB E9                              JMP GET_INPUT
2309  0000:00FF989C  8B 1D 98 02 00 00  COMPAIR_LAST:   MOV EBX,LAST_KEY               ;GET LAST KEY PRESSED
2310  0000:00FF98A2  3A C3                              CMP AL,BL
2311  0000:00FF98A4  74 DF                             JZ GET_INPUT                ;IF THE SAME, TRY AGAIN
2312  0000:00FF98A6  8A D8                             MOV BL,AL                   ;SAVE KEY IN BL
2313  0000:00FF98A8  B9 05 00 00 00                     MOV ECX,05H                 ;SCAN 5 TIMES FOR DEBOUNCE
2314  0000:00FF98AD  E8 6A FE FF FF     RETEST:         CALL SCAN                   ;SCAN KEYBOARD
2315  0000:00FF98B2  3A C3                              CMP AL,BL
2316  0000:00FF98B4  75 CF                             JNZ GET_INPUT              ;IF NOT THE SAME, START OVER
2317  0000:00FF98B6  E2 F5                              LOOP RETEST                  ;ELSE TEST 5 TIMES
2318  0000:00FF98B8  3C 28                              CMP AL,40                    ;IS  KEY VALID
2319  0000:00FF98BA  7F C9                              JG GET_INPUT                 ;IF NOT TRY AGAIN
2320  0000:00FF98BC  8A D8                              MOV BL,AL                    ;SAVE KEY PRESS
2321  0000:00FF98BE  67                                 DB ADDR_SZ
2322  0000:00FF98BF  89 1E 98 02                        MOV LAST_KEY,EBX
2323  0000:00FF98C3  E8 71 FF FF FF                     CALL BEEP                       ;BEEP
2324  0000:00FF98C8  3C 28                              CMP AL,40                    ;WAS SHIFT KEY PRESSED
2325  0000:00FF98CA  0F 84 17 00 00 00                  JZ SHIFT_SERVICE             ;IF SO SERVICE IT
2326  0000:00FF98D0  8B DD              L_5:            MOV EBX,EBP
2327  0000:00FF98D2  D7                                 XLAT                          ;GET KEY CODE
2328  0000:00FF98D3  8B 1D C8 02 00 00                  MOV EBX,MEM_VAL_3             ;CLEAR  SHIFT
2329  0000:00FF98D9  80 E7 00             DB 80H,E7H,00H
2330  0000:00FF98DC  67                                 DB ADDR_SZ
2331  0000:00FF98DD  89 1E C8 02                        MOV MEM_VAL_3,EBX
2332  0000:00FF98E1  5F                                 POP EDI
2333  0000:00FF98E2  5E                                 POP ESI
2334  0000:00FF98E3  5D                                 POP EBP
2335  0000:00FF98E4  59                                 POP ECX
2336  0000:00FF98E5  5B                                 POP EBX
2337  0000:00FF98E6  C3                                 RET
2338  0000:00FF98E7
2339                                    ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2340  0000:00FF98E7  8B 1D C8 02 00 00  SHIFT_SERVICE:  MOV EBX,MEM_VAL_3
2341  0000:00FF98ED  80 FB 00                 CMP BL,00
2342  0000:00FF98F0  74 DE                    JE L_5
2343                                    ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2344
2345
2346
2347  0000:00FF98F2  BE 2F 84 FF 00                     MOV ESI,OFFSET SHIFT_MSG        ;GET SHIFT_MSG ADDRESS
2348  0000:00FF98F7  BF EC 02 00 00                     MOV EDI,OFFSET DISPBF      ;GET DISPLAY BUFFER ADDRESS
2349  0000:00FF98FC  B9 11 00 00 00                     MOV ECX,00000011H          ;SET TO LOAD 17 CHARACTERS
2350
2351
2352                                    ;++++++++++++++++++++++++++++++++++++++++
2353  0000:00FF9901  80 FF 00                 DB 80H,FFH,00H
2354                                    ;++++++++++++++++++++++++++++++++++++++++
2355  0000:00FF9904  0F 84 11 00 00 00                  JZ SHIFT_IT                     ;NOT GO TO SHIFT_IT
2356                                    ;++++++++++++++++++++++++++++++++++
2357  0000:00FF990A  B7 00                    MOV BH,00H
2358                                    ;++++++++++++++++++++++++++++++++++
2359  0000:00FF990C  BD 2F 99 FF 00                     MOV EBP,OFFSET KEYTABLE         ;GET REGULAR KEY TABLE
2360  0000:00FF9911  BE 37 84 FF 00                     MOV ESI,OFFSET BLANK            ;GET BLANK MSG ADDRESS
2361  0000:00FF9916  E9 07 00 00 00                     JMP LD_DISPLAY
2362                                    ;+++++++++++++++++++++++++++++++++++++++++++++++
2363  0000:00FF991B  B7 FF              SHIFT_IT:          MOV BH,FFH
2364  0000:00FF991D  BD 58 99 FF 00           MOV EBP,00FF9958H
2365                                    ;+++++++++++++++++++++++++++++++++++++++++++++++
2366  0000:00FF9922
2367  0000:00FF9922  3E F3 A4                 LD_DISPLAY:     REP MOVSB DS:[EDI],CS:[ESI]     ;LOAD DISPLAY BUFFER
2368  0000:00FF9925  67                                       DB ADDR_SZ
2369  0000:00FF9926  89 1E C8 02                              MOV MEM_VAL_3,EBX               ;FIX SYS_FLAGS
2370  0000:00FF992A  E9 51 FF FF FF                           JMP UPDATE_DISPLAY              ;GET NEXT KEY
2371  0000:00FF992F
2372  0000:00FF992F  00                       KEYTABLE:      DB 00H                          ;NO KEY PRESSED
2373  0000:00FF9930
2374  0000:00FF9930  47                                       DB 47H                          ;USER KEY U7  1
2375  0000:00FF9931  0F                                       DB 0FH                          ;HEX KEY
2376  0000:00FF9932  0B                                       DB 0BH                          ;HEX KEY
2377  0000:00FF9933  07                                       DB 07H                          ;HEX KEY
2378  0000:00FF9934  03                                       DB 03H                          ;HEX KEY
2379  0000:00FF9935
2380  0000:00FF9935  43                                       DB 43H                          ;USERKEY U3 6
2381  0000:00FF9936  1F                                       DB 1FH                          ;ADDR
2382  0000:00FF9937  1B                                       DB 1BH                          ;TS32
2383  0000:00FF9938  17                                       DB 17H                          ;+
2384  0000:00FF9939  13                                       DB 13H                          ;-
2385  0000:00FF993A
2386  0000:00FF993A  46                                       DB 46H                          ;USER KEY U6   11
2387  0000:00FF993B  0E                                       DB 0EH                          ;HEX KEY
2388  0000:00FF993C  0A                                       DB 0AH                          ;HEX KEY
2389  0000:00FF993D  06                                       DB 06H                          ;HEX KEY
2390  0000:00FF993E  02                                       DB 02H                          ;HEX KEY
2391  0000:00FF993F
2392  0000:00FF993F  42                                       DB 42H                          ;USERKEY U2  16
2393  0000:00FF9940  1E                                       DB 1EH                          ;BYTE
2394  0000:00FF9941  1A                                       DB 1AH                          ;WORD
2395  0000:00FF9942  16                                       DB 16H                          ;DWORD
2396  0000:00FF9943  12                                       DB 12H                          ;CLEAR B.P.
2397  0000:00FF9944
2398  0000:00FF9944  45                                       DB 45H                          ;USER KEY U5   21
2399  0000:00FF9945  0D                                       DB 0DH                          ;HEX KEY
2400  0000:00FF9946  09                                       DB 09H                          ;HEX KEY
2401  0000:00FF9947  05                                       DB 05H                          ;HEX KEY
2402  0000:00FF9948  01                                       DB 01H                          ;HEX KEY
```

```
2403  0000:00FF9949
2404  0000:00FF9949  41                                    DB 41H                    ;USERKEY U1  26
2405  0000:00FF994A  1D                                    DB 1DH                    ;AUTO
2406  0000:00FF994B  19                                    DB 19H                    ;INSERT
2407  0000:00FF994C  15                                    DB 15H                    ;STEP
2408  0000:00FF994D  11                                    DB 11H                    ;RUN
2409  0000:00FF994E
2410  0000:00FF994E  44                                    DB 44H                    ;USER KEY U4
2411  0000:00FF994F  0C                                    DB 0CH                    ;HEX KEY
2412  0000:00FF9950  08                                    DB 08H                    ;HEX KEY
2413  0000:00FF9951  04                                    DB 04H                    ;HEX KEY
2414  0000:00FF9952  00                                    DB 00H                    ;HEX KEY
2415  0000:00FF9953
2416  0000:00FF9953  40                                    DB 40H                    ;USERKEY U0
2417  0000:00FF9954  1C                                    DB 1CH                    ;MOVE
2418  0000:00FF9955  18                                    DB 18H                    ;DNLOAD
2419  0000:00FF9956  14                                    DB 14H                    ;UPLOAD
2420  0000:00FF9957  10                                    DB 10H                    ;SHIFT (NOT RETURNED)
2421  0000:00FF9958
2422  0000:00FF9958  00                          SF_KEYTABLE:  DB 00H                ;NO KEY
2423  0000:00FF9959
2424  0000:00FF9959  57                                    DB 57H                    ;USER KEY SU7
2425  0000:00FF995A  3F                                    DB 3FH                    ;SHIHT F
2426  0000:00FF995B  3B                                    DB 3BH                    ;SHIFT B
2427  0000:00FF995C  37                                    DB 37H                    ;SHIFT 7
2428  0000:00FF995D  33                                    DB 33H                    ;SHIFT 3
2429  0000:00FF995E
2430  0000:00FF995E  53                                    DB 53H                    ;USERKEY SU3
2431  0000:00FF995F  2F                                    DB 2FH                    ;REL
2432  0000:00FF9960  2B                                    DB 2BH                    ;TS16
2433  0000:00FF9961  27                                    DB 27H                    ;+
2434  0000:00FF9962  23                                    DB 23H                    ;-
2435  0000:00FF9963
2436  0000:00FF9963  56                                    DB 56H                    ;USER KEY SU6
2437  0000:00FF9964  3E                                    DB 3EH                    ;SHIFT E
2438  0000:00FF9965  3A                                    DB 3AH                    ;SHIFT A
2439  0000:00FF9966  36                                    DB 36H                    ;SHIFT 6
2440  0000:00FF9967  32                                    DB 32H                    ;SHIFT 2
2441  0000:00FF9968
2442  0000:00FF9968  52                                    DB 52H                    ;USERKEY SU2
2443  0000:00FF9969  2E                                    DB 2EH                    ;SHIFT BYTE
2444  0000:00FF996A  2A                                    DB 2AH                    ;SHIFT WORD
2445  0000:00FF996B  26                                    DB 26H                    ;SHIFT DWORD
2446  0000:00FF996C  22                                    DB 22H                    ;SET B.P.
2447  0000:00FF996D
2448  0000:00FF996D  55                                    DB 55H                    ;USER KEY SU5
2449  0000:00FF996E  3D                                    DB 3DH                    ;SHIFT D
2450  0000:00FF996F  39                                    DB 39H                    ;SHIFT 9
2451  0000:00FF9970  35                                    DB 35H                    ;SHIFT 5
2452  0000:00FF9971  31                                    DB 31H                    ;SHIFT 1
2453  0000:00FF9972
2454  0000:00FF9972  51                                    DB 51H                    ;USERKEY SU1
2455  0000:00FF9973  2D                                    DB 2DH                    ;SHIFT AUTO
2456  0000:00FF9974  29                                    DB 29H                    ;DELETE
2457  0000:00FF9975  25                                    DB 25H                    ;SHIFT STEP
2458  0000:00FF9976  21                                    DB 21H                    ;SHIFT RUN
2459  0000:00FF9977
2460  0000:00FF9977  54                                    DB 54H                    ;USER KEY U4
2461  0000:00FF9978  3C                                    DB 3CH                    ;SHIFT C
2462  0000:00FF9979  38                                    DB 38H                    ;SHIFT 8
2463  0000:00FF997A  34                                    DB 34H                    ;SHIFT 4
2464  0000:00FF997B  30                                    DB 30H                    ;SHIFT 0
2465  0000:00FF997C
2466  0000:00FF997C  50                                    DB 50H                    ;USERKEY SU0
2467  0000:00FF997D  2C                                    DB 2CH                    ;SHIFT MOVE
2468  0000:00FF997E  28                                    DB 28H                    ;TAPE READ
2469  0000:00FF997F  24                                    DB 24H                    ;TAPE WRITE
2470  0000:00FF9980  20                                    DB 20H                    ;SHIFT (NOT RETURNED)
2471  0000:00FF9981
2472                                       ;-------------------------------------------------------------------------
2473                                       ;DISPLAY_INFO  - FORMATS THE CURRENT ADDRESS  AND THE  DATA AT THAT ADDRESS
2474                                       ;AND LOADS THEM INTO THE DISPLAY BUFFER. IF THE CURRENT ADDRESS IS NOT IN RAM
2475                                       ;OR  ROM  THE xxxxxxxx MESSAGE IS  DISPLAYED INSTEAD OF THE DATA. CHARACTER
2476                                       ;8 OF THE DISPLAY (NORMALY AN ARROW) IS NOT CHANGED BY THIS ROUTINE AND
2477                                       ;MUST BE UPDATED  BEFORE CALLING THIS ROUTINE
2478                                       ;-------------------------------------------------------------------------
2479  0000:00FF9981
2480  0000:00FF9981  50                          DISPLAY_INFO:  PUSH EAX
2481  0000:00FF9982  51                                         PUSH ECX
2482  0000:00FF9983  56                                         PUSH ESI
2483  0000:00FF9984  57                                         PUSH EDI
2484  0000:00FF9985
2485  0000:00FF9985  BE A0 02 00 00                             MOV ESI,OFFSET CURRENT_ADDRESS ;FORMAT CURRENT ADDRESS
2486  0000:00FF998A  BF F3 02 00 00                              MOV EDI,OFFSET DISPBF+7       ;FOR DISPLAY
2487  0000:00FF998F  B9 04 00 00 00                              MOV ECX,00000004H
2488  0000:00FF9994  E8 4F 00 00 00                              CALL FORMAT
2489  0000:00FF9999
2490  0000:00FF9999  BE 59 84 FF 00                              MOV ESI,OFFSET X_MSG          ;LOAD xxxxxxxx MESSAGE
2491  0000:00FF999E  BF F5 02 00 00                              MOV EDI,OFFSET DISPBF+9        ;INTO  DISPLAY BUFFER
2492  0000:00FF99A3  B9 08 00 00 00                              MOV ECX,00000008H
2493  0000:00FF99A8  3E F3 A4                                    REP MOVSB DS:[EDI],CS:[ESI]
2494  0000:00FF99AB
2495  0000:00FF99AB  8B 35 A0 02 00 00                           MOV ESI,CURRENT_ADDRESS       ;GET CURRENT ADDRESS
2496  0000:00FF99B1  8B 0D 1C 02 00 00                           MOV ECX,A_MODE                ;GET DATA MODE
2497
2498                                       ;++++++++++++++++++++++++++++++++++++++++++++++
2499  0000:00FF99B7  8B 1D 98 00 00 00             MOV EBX,COPRO_NOT_AVAL
2500  0000:00FF99BD  43                            INC EBX
```

```
2501                                    ;++++++++++++++++++++++++++++++++++++++++++++
2502
2503
2504  0000:00FF99BE  2B D9                                  SUB EBX,ECX             ;SUBTRACT DATA_MODE
2505  0000:00FF99C0  3B F3                                  CMP ESI,EBX             ;IF CURRENT_ADDRESS IS LESS OR
2506  0000:00FF99C2  0F 8E 11 00 00 00                      JLE DISPLAY_DATA        ;EQUAL, THEN  IT IS IN RAM
2507  0000:00FF99C8  81 FE 00 80 FF 00                      CMP ESI,00FF8000H       ;ADDRESS OF FIRST ROM BYTE
2508  0000:00FF99CE  0F 8D 05 00 00 00                      JGE DISPLAY_DATA    ;IF CURREN_ADDRESS IS GREATER  OR
2509                                                                            ;IT IS  IN ROM
2510  0000:00FF99D4  E9 0A 00 00 00                         JMP DISPLAY_DONE        ;ELSE DATA AS xxxxxxxx
2511  0000:00FF99D9
2512  0000:00FF99D9  BF FC 02 00 00           DISPLAY_DATA:  MOV EDI,OFFSET DISPBF+16        ;FORMAT DATA
2513  0000:00FF99DE  E8 05 00 00 00                         CALL FORMAT
2514  0000:00FF99E3
2515  0000:00FF99E3  5F                       DISPLAY_DONE:  POP EDI
2516  0000:00FF99E4  5E                                     POP ESI
2517  0000:00FF99E5  59                                     POP ECX
2518  0000:00FF99E6  58                                     POP EAX
2519  0000:00FF99E7  C3                                     RET
2520  0000:00FF99E8
2521                                    ;-------------------------------------------------------------------------
2522                                    ;FORMAT - FORMATS HEX DATA FOR THE DISPLAY. ESI  SHOULD POINT TO FIRST BYTE
2523                                    ;OF DATA TO BE FORMATTED, EDI SHOULD POINT TO THE FIRST BYTE (RIGHTMOST DIGIT)
2524                                    ;OF THE DISPLAY TO BE LOADED AND ECX SHOULD  CONTAIN THE  # OF BYTES  TO BE
2525                                    ;FORMATTED. SUCESSIVELY HIGHER BYTES ARE THEN FORMATTED AND LOADED INTO
2526                                    ;SUCESSIVELY LOWER BYTES IN THE DISPLAY BUFFER UNTIL ECX BYTES HAVE BEEN
2527                                    ;FORMATTED. EACH HEX DATA BYTE TO BE FORMATTED RESULTS IN TWO CHARACTERS
2528                                    ;BEING LOADED INTO THE DISPLAY BUFFER (USING TWO BYTES OF THE DISPLAY BUFFER)
2529                                    ;THUS TWO RIGHTMOST CHRACTERS OF THE  DISPLAY WILL REPRESENT THE LOWER MEMORY
2530                                    ;BYTE FORMATTED AND EACH  PAIR OF CHARACTERS TO THE  LEFT REPRESENTS THE NEXT
2531                                    ;HIGHER BYTE OF MEMORY
2532                                    ;-------------------------------------------------------------------------
2533  0000:00FF99E8
2534  0000:00FF99E8  53                       FORMAT:        PUSH EBX
2535  0000:00FF99E9  52                                     PUSH EDX
2536  0000:00FF99EA
2537  0000:00FF99EA  BB 1F 84 FF 00                    MOV EBX,OFFSET HEXCODES ;GET DISPLAY CODES FOR HEX  DIGITS
2538  0000:00FF99EF
2539  0000:00FF99EF  8A 06                    NEXT_BYTE:     MOV AL,[ESI]                    ;GET DATA BYTE IN AL
2540  0000:00FF99F1  8A D0                                   MOV DL,AL                       ;SAVE  IT IN DL
2541  0000:00FF99F3  24 0F                                   AND AL,0FH                      ;CLEAR HIGH 4  BITS
2542  0000:00FF99F5  D7                                      XLAT                    ;GET HEX CODE  FOR LOW  4 BITS
2543  0000:00FF99F6  88 07                                   MOV [EDI],AL            ;PUT IT IN DISPLAY BUFFER
2544  0000:00FF99F8
2545  0000:00FF99F8  8A C2                                   MOV AL,DL              ;RESTORE DATA
2546  0000:00FF99FA  C1 C8 04                                ROR EAX,04H            ;ROL HIGH 4 BITS  INTO POSITION
2547  0000:00FF99FD  24 0F                                   AND AL,0FH
2548  0000:00FF99FF  D7                                      XLAT                   ;GET  HEX CODE FOR HIGH 4 BITS
2549  0000:00FF9A00  4F                                      DEC EDI                ;DECREMENT DISPLAY POINTER
2550  0000:00FF9A01  88 07                                   MOV [EDI],AL      ;LOAD HEX CODE  INTO DISPLAY BUFFER
2551  0000:00FF9A03
2552  0000:00FF9A03  4F                                      DEC EDI                    ;DECREMENT DISPLAY POINTER
2553  0000:00FF9A04  46                                      INC ESI                    ;INCREMENT DATA  POINTER
2554  0000:00FF9A05  E2 E8                                   LOOP NEXT_BYTE             ;ECX CONTAINS  # OF BYTES
2555                                                                                    ;TO BE  FORMATTED
2556  0000:00FF9A07  5A                                      POP EDX
2557  0000:00FF9A08  5B                                      POP EBX
2558  0000:00FF9A09  C3                                      RET
2559                                    ;-------------------------------------------------------------------------
2560                                    ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2561  0000:00FF9A0A  E8 36 00 00 00     SUB_1:     CALL 036H
2562  0000:00FF9A0F  3C 13                         CMP AL,13H
2563  0000:00FF9A11  0F 84 12 00 00 00             JZ EX_1
2564  0000:00FF9A17  E8 44 00 00 00                CALL 044H
2565  0000:00FF9A1C  3C 13                         CMP AL,13H
2566  0000:00FF9A1E  74 EA                         DB 74H,EAH              ;JZ LABEL
2567  0000:00FF9A20  E8 56 00 00 00                CALL 056H
2568  0000:00FF9A25  3C 13                         CMP AL,13H
2569  0000:00FF9A27  74 EE                         DB 74H,EEH               ;JZ LABEL
2570  0000:00FF9A29  C3                 EX_1:      RET
2571  0000:00FF9A2A  52                            PUSH EDX
2572  0000:00FF9A2B  55                            PUSH EBP
2573  0000:00FF9A2C  56                            PUSH ESI
2574  0000:00FF9A2D  BA F0 FF FF 00                MOV EDX,00FFFFF0H
2575  0000:00FF9A32  BD A4 02 00 00                MOV EBP,OFFSET DATA_MODE
2576  0000:00FF9A37  BE 61 84 FF 00                MOV ESI,00FF8461H
2577  0000:00FF9A3C  E8 55 00 00 00                CALL 055H
2578  0000:00FF9A41  5E                            POP ESI
2579  0000:00FF9A42  5D                            POP EBP
2580  0000:00FF9A43  5A                            POP EDX
2581  0000:00FF9A44  C3                            RET
2582  0000:00FF9A45  52                            PUSH EDX
2583  0000:00FF9A46  55                            PUSH EBP
2584  0000:00FF9A47  56                            PUSH ESI
2585  0000:00FF9A48  BA F0 FF FF 00                MOV EDX,00FFFFF0H
2586  0000:00FF9A4D  BD A8 02 00 00                MOV EBP,OFFSET START_ADDRESS
2587  0000:00FF9A52  BE 69 84 FF 00                MOV ESI,00FF8469H
2588  0000:00FF9A57  E8 3A 00 00 00                CALL 03AH
2589  0000:00FF9A5C  5E                            POP ESI
2590  0000:00FF9A5D  5D                            POP EBP
2591  0000:00FF9A5E  5A                            POP EDX
2592  0000:00FF9A5F  C3                            RET
2593  0000:00FF9A60  52                            PUSH EDX
2594  0000:00FF9A61  55                            PUSH EBP
2595  0000:00FF9A62  56                            PUSH ESI
2596  0000:00FF9A63  BA F0 FF FF 00                MOV EDX,00FFFFF0H
2597  0000:00FF9A68  BD AC 02 00 00                MOV EBP,OFFSET END_ADDRESS
2598  0000:00FF9A6D  BE 71 84 FF 00                MOV ESI,00FF8471H
```



```
0000:00FF9A72  E8 1F 00 00 00              CALL 01FH
0000:00FF9A77  5E                          POP ESI
0000:00FF9A78  5D                          POP EBP
0000:00FF9A79  5A                          POP EDX
0000:00FF9A7A  C3                          RET
0000:00FF9A7B  52                          PUSH EDX
0000:00FF9A7C  55                          PUSH EBP
0000:00FF9A7D  56                          PUSH ESI
0000:00FF9A7E  BA F0 FF FF 00              MOV EDX,00FFFFF0H
0000:00FF9A83  BD E8 02 00 00              MOV EBP,OFFSET MOVE_INFO
0000:00FF9A88  BE BD 84 FF 00              MOV ESI,00FF84BDH
0000:00FF9A8D  E8 04 00 00 00              CALL 04H
0000:00FF9A92  5E                          POP ESI
0000:00FF9A93  5D                          POP EBP
0000:00FF9A94  5A                          POP EDX
0000:00FF9A95  C3                          RET

                                    ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

0000:00FF9A96  53                   GET_USER:       PUSH EBX
0000:00FF9A97  51                                   PUSH ECX
0000:00FF9A98  57                                   PUSH EDI
0000:00FF9A99  33 DB                  XOR EBX,EBX
0000:00FF9A9B  BF EC 02 00 00                       MOV EDI,OFFSET DISPBF
0000:00FF9AA0  B9 08 00 00 00                       MOV ECX,00000008H
0000:00FF9AA5  3E F3 A4                             REP MOVSB DS:[EDI],CS:[ESI]
0000:00FF9AA8  C6 07 7E                             MOV [EDI],BYTE PTR 7EH
0000:00FF9AAB  B9 04 00 00 00       RE_FORMAT:      MOV ECX,00000004H
0000:00FF9AB0  8B F5                                MOV ESI,EBP
0000:00FF9AB2  BF FC 02 00 00                       MOV EDI,OFFSET DISPBF+16
0000:00FF9AB7  E8 2C FF FF FF                       CALL FORMAT
0000:00FF9ABC  E8 B3 FD FF FF       GET_NEW_DIGIT:  CALL KEYBOARD
0000:00FF9AC1  3C 13                                CMP AL,13H
0000:00FF9AC3  0F 84 19 00 00 00                    JZ GOT_ADDR
0000:00FF9AC9  3C 17                                CMP AL,17H
0000:00FF9ACB  0F 84 11 00 00 00                    JZ GOT_ADDR
0000:00FF9AD1  3C 0F                                CMP AL,0FH
0000:00FF9AD3  7F E7                                JG GET_NEW_DIGIT
0000:00FF9AD5  C1 C3 04                             ROL EBX,04H
0000:00FF9AD8  23 DA                                AND EBX,EDX
0000:00FF9ADA  0A D8                                OR BL,AL
                                    ;++++++++++++++++++++
0000:00FF9ADC  8B FD                        MOV EDI,EBP
                                    ;++++++++++++++++++++
0000:00FF9ADE  89 1F                                MOV [EDI],EBX
0000:00FF9AE0  EB C9                                JMP RE_FORMAT
0000:00FF9AE2
0000:00FF9AE2  5F                   GOT_ADDR:       POP EDI
0000:00FF9AE3  59                                   POP ECX
0000:00FF9AE4  5B                                   POP EBX
0000:00FF9AE5  C3                                   RET
0000:00FF9AE6
0000:00FF9AE6
0000:00FF9AE6

                                    ;*************************************************************************
                                    ;*                                                                       *
                                    ;*          ADDITIONAL CONSTANTS USED BY THE MONITOR                     *
                                    ;*                                                                       *
                                    ;*************************************************************************

0000:00FF9AE6
                    0000:00000066           DATA_SZ:        EQU 66H
                    0000:00000067           ADDR_SZ:        EQU 67H
0000:00FF9AE6
                    0000:000000FF           TOP_ROM:        EQU  00FFH
0000:00FF9AE6
                                        ;8255  PORT ADDRESSES
                    0000:0000E000           PORTA_L:        EQU 00E000H
                    0000:0000E002           PORTB_L:        EQU 00E002H
                    0000:0000E004           PORTC_L:        EQU 00E004H
                    0000:0000E006           CONTR_L:        EQU 00E006H
0000:00FF9AE6
                    0000:0000E001           PORTA_H:        EQU 00E001H
                    0000:0000E003           PORTB_H:        EQU 00E003H
                    0000:0000E005           PORTC_H:        EQU 00E005H
                    0000:0000E007           CONTR_H:        EQU 00E007H
0000:00FF9AE6


                    0000:0000F000             S_PORT_0: EQU 00F000H
                    0000:0000F001             S_PORT_1: EQU 00F001H
                    0000:0000F002             S_PORT_2: EQU 00F002H
                    0000:0000F003             S_PORT_3: EQU 00F003H
                    0000:0000F004             S_PORT_4: EQU 00F004H
                    0000:0000F005             S_PORT_5: EQU 00F005H
                    0000:0000F006             S_PORT_6: EQU 00F006H
                    0000:0000F007             S_PORT_7: EQU 00F007H
                    0000:0000F008             S_PORT_8: EQU 00F008H
                    0000:0000F009             S_PORT_9: EQU 00F009H
                    0000:0000F00A             S_PORT_A: EQU 00F00AH
                    0000:0000F00B             S_PORT_B: EQU 00F00BH
                    0000:0000F00C             S_PORT_C: EQU 00F00CH
                    0000:0000F010             S_PORT_10: EQU 00F010H


0000:00FFFFF0                               ORG 00FFFFF0H
0000:00FFFFF0
0000:00FFFFF0  90                           NOP
0000:00FFFFF1  E9 0E 80 FF 00               DB E9H,0EH,80H,FFH,00H
```

```
2697   0000:00FFFFF6
2698   0000:00FFFFF6                                   END

          Lines Assembled :  2698            Assembly Errors :  32
```


URDA®, INC. PRODUCT AND PRICE LIST

Effective September 1, 1990

Hardware

Cables

Software

Special Prices on Complete Development Systems

PRICE LIST (PASTE IN)

URDA®, INC. µLAB™ PRODUCT LIST AND PRICES

EFFECTIVE SEPTEMBER 1, 1990

Call URDA® for current prices and quantity discounts. Inquiries invited from qualified distributors.

HARDWARE

NO.	PRODUCT	DESCRIPTION	List
H1	P68000 µLab™ Notebook Computer™	16 Bit Microprocessor Development System	320.00
H2	P8086 µLab™ Notebook Computer™	16 Bit Microprocessor Development System	320.00
H3	P68020/68881 µLab™ Upgrade[1]	32 Bit Microprocessor Development System for P68000	320.00
H4	PADC-DAC-8 µLab™	ADC and DAC Peripheral for P68000 and P8086	320.00
H5	P32010 µLab™	Auxiliary Digital Signal Processor for P68000 µLab™	320.00
H6	68020 CPU Chip	For P68020/68881 µLab™ (with MC68020 User's Manual)	175.00
H7	68881 Coprocessor Chip	For P68020/68881 µLab™ (with MC68881 User's Manual)	145.00
H8	PWWEB-0 µLab™ Wire Wrap Kit	For P68000 and P8086 µLab™	30.00
H9	PWWEB-1 µLab™ Wire Wrap Kit	For P68000 and P8086 µLab™ (With User's Manual)	70.00
H10	PWWEB-2 µLab™ Wire Wrap Kit	For P68000 and P8086 µLab™ (With User's Manual)	90.00
H11	P68681 µLab™	Communications Interface for P68000 and P68020	130.00
H12	P8251A µLab™	Communications Interface for P8086	130.00
H13	P8087 µLab™ (Without 8087 Chip)	Coprocessor Peripheral for P8086	100.00
H14	PBURN µLab™	EPROM Burner for P68000 and P8086	220.00
H15	PMEM(68000) µLab™ (without Chips)	Memory Expansion Kit for the P68000 µLab™	50.00
H16	8087 Coprocessor Chip	For P8087 µLab™	140.00
H17	P68030 µLab™ Notebook Computer™2	32 Bit Microprocessor Development System - Complete 2 Serial Ports, 32K RAM, 16K EPROM, LCD Display	995.00
H18	SDK-85 System Development Kernel	8 Bit Microprocessor Development System - Complete	285.00
H19	SDK-86 System Development Kernel	16 Bit Microprocessor Development System - Complete	450.00
H20	SDK-386™* Sys Development Kernel3	32 Bit Microprocessor Development System - Complete 2 Serial Ports, 32K RAM, 16K EPROM, LCD Display	995.00

CABLES

C1	PACC-1 µLab™	Tape Cable Set (2 Cables) for Tape Recorder	15.00
C2	PACC-2 µLab™	40 Conductor Ribbon Cable (1 Cable)	15.00
C3	PACC-3 µLab™	50 Conductor Ribbon Cable (1 Cable)	15.00
C4	PACC-4 µLab™	DB9M TO DB25F Cable (Mac® 512K to P68681 µLab™)	25.00
C5	PACC-5 µLab™	DB25F to DB25M Cable (PC TO P68681 µLab™)	20.00
C6	PACC-6 µLab™	50 Conductor Cable (3 50 Pin Connectors Pin to Pin for P68000 to P68681 to P68020, etc.)	20.00

SOFTWARE

S1	PCOM-MAC µLab™	Communications Software for the MACINTOSH®	25.00
S2	PCOM-PC µLab™	Communications Software for IBM PC® and Compatibles	25.00
S3	PASM-8086 µLab™	P8086 Assembler for IBM PC® and Compatibles	99.95
S4	PASM-68000 µLab™	P68000 Cross Assembler for IBM PC® and Compatibles	49.95
S5	PASM-68020 µLab™	P680X0 Cross Assembler for IBM PC® and Compatibles	149.95
S6	PSHAREWARE µLab™	3 5.25" Disks of SHAREWARE including an editor, a terminal program and an assembler all for IBM PC®.	15.00
S7	PASM-32010 µLab™	P32010 µLab™ Assembler for MS-DOS Computers	99.95
S8	PASM-80386 µLab™	SDK-386™ Cross Assembler for MS-DOS Computers	199.95
S9	PASM-8085 µLab™	URDA SDK-85 Cross Assembler for MS-DOS Computers	100.00
S10	PCMPLR-C µLab™	P68000 C Cross Compiler for MS-DOS Computers	150.00

SPECIAL PRICES ON COMPLETE DEVELOPMENT SYSTEMS

NO.	NAME	CONTENTS (By Number Above)	List
CS1	P68000 µLab™ -CS	H1, H11, C3 (2), C5, S4	539.00
CS2	P68000/68020/68881 µLab™ -CS	H1, H3, H6, H7, H11, C5, C6 (2), S5	1274.00
CS3	P68000/32010 µLab™ -CS	H1, H5, H11, C5, C6(2), S4, S7	960.00
CS4	P8086 µLab™ -CS	H2, H12, C3(2), C5, S3	588.00
CS5	P68030 µLab™ - CS[2]	H17, C3(3), S1, S2, S5, S6	1230.00
CS6	SDK-386™ - CS[3]	H20, C3(3), S1, S2, S6, S8	1279.00

*386 is a trademark of Intel Corporation. Used by permission

[1] Does not include MC68020 and MC68881 chips which may be purchased separately from URDA®.

[2] Does not include MC68030 and MC68882 chips which may be purchased separately from URDA®.

[3] Does not include 80386 and 80387 chips which may be purchased separately from URDA®.

Note 1: Prices are subject to change without notice. For most recent prices, call URDA®, Inc at the number listed.

Note 2: All prices are U. S. DOLLARS for each unit. Additional Shipping and Handling charges are applicable. Call URDA®, Inc. for details.

Note 3: Hardware Items 1, 2, 3, 4 and 5 above may be mixed or matched to obtain quantity pricing.

Note 4: For detailed specifications for any item, see the Catalog Sheets available from URDA®, Inc.

Note 5: Software Items 1, 2 and 7 are URDA®, Inc. proprietary software for use with µLAB™ hardware and software. Items 3, 4, 5, 8, 9 and 10 are professionally developed software either licensed to URDA®, Inc., or retailed by URDA®, Inc. The fee for Item 6 is for media and distribution costs only. You must register and pay a licensing fee to the author of the shareware program for each program you decide to continuously use. These Shareware Programs have been hand selected by URDA®, Inc. personnel for their utility and usefulness in working with µLAB™ hardware and software and are licensed to URDA®, Inc. for distribution.

ADDITIONAL INFORMATION

To obtain additional information, contact URDA®, Inc., 4516 Henry Street, Suite #407, Pittsburgh, PA 15213, or

CALL 1-800-338-0517 or 412-683-8732.

URDA